University of Tennessee, Knoxville
Trace: Tennessee Research and Creative Exchange

Doctoral Dissertations | Graduate School

5-2015

# Hybrid K-edge Densitometry as a Method for Materials Accountancy Measurements in Pyrochemical Reprocessing

Matthew Tyler Cook
*University of Tennessee - Knoxville*, mcook4@vols.utk.edu

## Recommended Citation

Cook, Matthew Tyler, "Hybrid K-edge Densitometry as a Method for Materials Accountancy Measurements in Pyrochemical Reprocessing. " PhD diss., University of Tennessee, 2015.
https://trace.tennessee.edu/utk_graddiss/3329

This Dissertation is brought to you for free and open access by the Graduate School at Trace: Tennessee Research and Creative Exchange. It has been accepted for inclusion in Doctoral Dissertations by an authorized administrator of Trace: Tennessee Research and Creative Exchange. For more information, please contact trace@utk.edu.

To the Graduate Council:

I am submitting herewith a dissertation written by Matthew Tyler Cook entitled "Hybrid K-edge Densitometry as a Method for Materials Accountancy Measurements in Pyrochemical Reprocessing." I have examined the final electronic copy of this dissertation for form and content and recommend that it be accepted in partial fulfillment of the requirements for the degree of Doctor of Philosophy, with a major in Nuclear Engineering.

Steven E. Skutnik, Major Professor

We have read this dissertation and recommend its acceptance:

Howard L. Hall, Joseph R. Stainback, Lawrence Heilbronn, James Ostrowski

Accepted for the Council:
Dixie L. Thompson

Vice Provost and Dean of the Graduate School

(Original signatures are on file with official student records.)

# Hybrid K-edge Densitometry as a Method for Materials Accountancy Measurements in Pyrochemical Reprocessing

A Dissertation Presented for the

Doctor of Philosophy

Degree

The University of Tennessee, Knoxville

Matthew Tyler Cook

May 2015

© by Matthew Tyler Cook, 2015
All Rights Reserved.

*This work is dedicated to my wife Stephanie, my parents, family, friends, mentors, co-workers and all those who helped my along the way.*

# Acknowledgements

This work has been in the making for many years and would not be possible without those around me. My wife Stephanie has supported me in the pursuit of my PhD. and without her love and dedication I would be lost. I would like to thank my parents and sister as well. Without their support this would not have been possible.

My advisors and professors have guided my efforts throughout my time at the University of Tennessee. Steven Skutnik and Howard Hall have been instrumental in not only this research effort but also have supported my work in other projects that have been instrumental in developing my skills as an engineer.

My coworkers have been instrumental in this process by providing fun and friendly work environment. Michael Willis, John Auxier, and Andrew Giminaro have made working on this research and many other projects infinitely more enjoyable.

Thanks also go to the team at Oak Ridge National Laboratory for providing data that made this project possible.

*Not that I have already obtained all this, or have already arrived at my goal, but I press on to take hold of that for which Christ Jesus took hold of me. Brothers and sisters, I do not consider myself yet to have taken hold of it. But one thing I do: Forgetting what is behind and straining toward what is ahead, I press on toward the goal to win the prize for which God has called me heavenward in Christ Jesus.*

*Philippians 3:12-14*

# Abstract

Pyrochemical reprocessing is a proven method to recover useful fissile material from spent nuclear fuel. The process requires high temperatures and an inert atmosphere thus complicating the prospect of making materials accountancy measurements. Development of a measurement method for materials accountancy measurements has become necessary since pyroprocessing is receiving more attention as a possible compliment to aqueous reprocessing methods. If pyroprocessing is to be adapted from the engineering scale to a commercially viable reprocessing method a comprehensive safeguards measurement method and strategy must be developed.

Hybrid k-edge densitometry (HKED) has been applied to aqueous reprocessing measurements in commercial facilities. This method relies on a tuned beam of x-rays to bombard a sample. X-rays at an element's k-edge, or the binding energy of the k-shell electrons, will be preferentially absorbed leading to a transmission drop in the beam. Electrons from a higher energy level fill the vacancies resulting from this absorption. This results in the emission of characteristic x-rays that are unique to a given element. The fusion of these two measurement methods indicate the density and elemental composition of the sample.

A MCNP model of a commercial HKED instrument was developed to perform scoping studies in support of this measurement method's application to pyroprocessing. Development of a strategy of applying HKED required the prediction of how the instrument will respond to samples from a pyroprocess. Aqueous samples were measured in the instrument at Oak Ridge National Laboratory and compared to

the results from the model. Some discrepancies were identified and are attributed to inconsistencies in both the modeled x-ray spectrum and MCNP photon libraries, however the model does effectively represent the characteristic x-rays and k-edge drop.

Several notional samples from a pyroprocess, such as molten salt solutions, voloxidation powders, liquid cathodes, and metallic strips were modeled to determine the system's response. Simulations of mixtures containing uranium and thorium (in place of plutonium) were completed to determine the feasibility of decreasing sample preparation cost without sacrificing sample characteristics. Finally, an active gamma-ray emitting sample was modeled to determine if x-ray fluorescence could be self-induced by the sample.

# Table of Contents

**1 Introduction** **1**

**2 Pyrochemical Reprocessing of Spent Nuclear Fuel** **3**

2.1 Overview . . . . . 3

2.2 Process Description . . . . . 4

2.2.1 Electrorefining . . . . . 6

2.2.2 Salt Regeneration and Impurity Occlusion . . . . . 8

2.2.3 Final Waste Form . . . . . 10

**3 Reprocessing Safeguards** **12**

3.1 Aqueous Reprocessing . . . . . 12

3.2 Measurement Methods . . . . . 15

**4 Hybrid K-edge Densitometry** **18**

4.1 Overview . . . . . 18

4.2 Instrument Description . . . . . 21

**5 Monte Carlo Modeling** **23**

5.1 HKED Model . . . . . 23

5.1.1 Overview . . . . . 23

5.1.2 Variance Reduction Methods . . . . . 27

5.1.3 Photon Library Comparison . . . . . 30

5.1.4 Pulse Height Spectrum Shift . . . . . 32

**6 Model Validation** **35**

6.1 Validation Simulations . . . . . 35

6.1.1 K X-ray Fluorescence Cases . . . . . 36

6.1.2 K X-ray Peak Intensity Analysis . . . . . 47

6.1.3 K-edge Transmission Cases . . . . . 54

6.1.4 K-edge Continuum Analysis . . . . . 63

6.2 X-ray Intensity and K-edge Response . . . . . 67

**7 Pyrochemical Cases** **71**

7.1 Molten Salt Solutions . . . . . 71

7.1.1 X-ray Fluorescence . . . . . 72

7.1.2 K-edge Transmission . . . . . 74

7.2 Dense Pyrochemical Samples . . . . . 75

7.2.1 X-ray Fluorescence . . . . . 77

7.2.2 K-edge Transmission . . . . . 80

7.3 Surrogate Samples . . . . . 83

7.3.1 X-ray Fluorescence . . . . . 84

7.3.2 K-edge Transmission . . . . . 89

7.4 Active Sample Source Term . . . . . 95

**8 Future Work** **97**

8.1 MCNP Photon Library Inconsistency . . . . . 97

8.2 X-ray Tube Characterization . . . . . 97

8.3 Gadolinium Filter Modifications . . . . . 98

8.4 Further Burnup and Cooling Times . . . . . 98

8.5 In-Situ Measurements . . . . . 98

8.6 $^{238}$Pu and Medical Isotope Process Measurements . . . . . 99

**9 Conclusions** **100**

**Bibliography** **103**

## Appendix 109

**A MCNP Input Decks** **110**

A.1 XRF: Stage 1 . . . . . 110

A.1.1 Core Input Deck . . . . . 110

A.1.2 Geometry . . . . . 112

A.1.3 Variance Reduction . . . . . 122

A.1.4 Problem Tallies . . . . . 123

A.1.5 Options . . . . . 125

A.2 XRF: Stage 2 . . . . . 126

A.2.1 Core Input Deck . . . . . 126

A.2.2 Geometry . . . . . 128

A.2.3 Variance Reduction . . . . . 138

A.2.4 Problem Tallies . . . . . 139

A.2.5 Options . . . . . 140

A.3 KED: Stage 1 . . . . . 141

A.3.1 Core Input Deck . . . . . 141

A.3.2 Geometry . . . . . 143

A.3.3 Variance Reduction . . . . . 153

A.3.4 Problem Tallies . . . . . 155

A.3.5 Options . . . . . 156

A.4 KED: Stage 2 . . . . . 157

A.4.1 Core Input Deck . . . . . 157

A.4.2 Geometry . . . . . 159

A.4.3 Variance Reduction . . . . . 169

A.4.4 Problem Tallies . . . . . 170

A.4.5 Options . . . . . 171

**B Sample Compositions** **173**

B.1 Aqueous Uranium Solutions . . . . . 173

B.2 Aqueous Uranium and Plutonium Solutions . . . . . 175

B.3 Salt Solutions . . . . . 176

B.4 Surrogate Salt Solutions . . . . . 178

B.5 Voloxidation Powder . . . . . 179

B.6 Liquid Cathode . . . . . 181

**C Analysis Tools** **182**

C.1 Hybrid K-edge Python Analysis Tools . . . . . 182

C.1.1 Aqueous Samples . . . . . 182

C.1.2 Pyrochemical Samples . . . . . 228

C.1.3 Surrogate Samples . . . . . 259

C.1.4 Complex Samples . . . . . 290

C.1.5 Active Samples . . . . . 321

C.2 Hybrid K-edge Results Plotting Script . . . . . 352

C.3 X-ray Spectrum Stretch Tool (X2ST) . . . . . 367

**Vita** **370**

# List of Tables

4.1 X-ray emission energies and K-edges for uranium and plutonium (1). 20

5.1 Parameters and confidence bounds for the polynomial fit to the energy offset. . . . . 33

6.1 Linear fit parameters for the K x-ray intensity offset. . . . . 54

6.2 Linear fit parameters for the K-edge continuum drop difference for uranium. . . . . 66

7.1 Pyrochemical sample attenuation coefficients and thicknesses. . . . . 76

# List of Figures


2.1 A pyrochemical process flowsheet for processing ceramic power reactor fuel. . . . . 5

2.2 Material flow within a generic electrorefiner without a liquid cadmium cathode. . . . . 6

3.1 Flowsheet illustrating the flow of material in PUREX reprocessing (2). 12

3.2 Countercurrent-multistage continuous flow through multiple extraction columns (2). . . . . 13

3.3 Typical liquid-liquid extraction column (3). . . . . 14

3.4 Typical layout of pressure sensors in a pneumatic dip tube measurement setup (4). . . . . 16

4.1 Physical processes behind HKED measurements. . . . . 19

4.2 Photon absorption edges for uranium. . . . . 19

4.3 HKED system without outer lead shielding. Both XRF (left) and KED (right) component beamlines are exposed and germanium detectors have been removed . . . . . 22

5.1 X-ray spectrum generated using SpekCalc. Maximum accelerator voltage was set to 150 keV. . . . . 25

5.2 Left: HKED geometry generated by MCNP. Right: MCNP mesh tally showing the particle population in the system geometry. Note that the photon population is vastly greater in the K-edge beamline. . . . . 26

5.3 3D rendering of the HKED model in VisEd. The XRF beamline is shown on the left and the KED beamline on the right. . . . . 26
5.4 Pulse height spectrum of 323.7 $^{g}/_{L}$ uranium using *mcplib84* and *mcplib12*. $K_{\beta}$ peaks are not well represented at between 110 and 115 keV using *mcplib84*, however the $K_{\beta}$ peaks are correctly modeled using *mcplib12*. . . . . 31
5.5 Calculated energy offset for both $K_{\alpha 1}$ and $K_{\alpha 2}$ photon emissions from a variety of simulations in MCNP compared to NIST XRF emission energies. The polynomial fit to these data sets is also shown with an $R^2$ value of 0.9967. . . . . 32
5.6 Uncorrected and corrected pulse height spectrum showing MCNP energy shift. Data presented in red represents the modeled uncorrected spectrum while the corrected spectrum is shown in green. Measured data is shown in blue. Higher deviations from the measured data are present at between 94 and 99 keV and 110 and 115 keV which correspond the $K_{\alpha}$ and $K_{\beta}$ peaks, respectively. . . . . 33

6.1 XRF pulse height spectrum of 1.072 $^{g}/_{L}$ uranium solution and residuals. Large differences between the model and the measured data are present at energies below 80 keV. The model does accurately model the $K_{\alpha 1}$ and $K_{\alpha 2}$ peak intensities but not the area between the $K_{\beta}$ peaks. The difference here is due to the lack of photons from the x-ray source. . . . . 36
6.2 XRF pulse height spectrum of 5.459 $^{g}/_{L}$ uranium solution and residuals. As in the 1.072 $^{g}/_{L}$ case the model does not capture the gadolinium peaks between 40 and 50 keV. The $K_{\alpha 1}$ and $K_{\alpha 2}$ peaks are correctly modeled, however the energy ranges between the peaks are under represented but not as much as the 1.072 $^{g}/_{L}$ case. . . . . 37

6.3 XRF pulse height spectrum of 15.895 $^{g}/_{L}$ uranium solution and residuals. Lower energies are still over represented in the model, however the energy ranges between the x-ray peaks are closer to the measured data. $K_{\alpha}$ and $K_{\beta}$ peak shapes are accurately represented. . 38
6.4 XRF pulse height spectrum of 48.12 $^{g}/_{L}$ uranium solution and residuals. Accuracy of gadolinium x-ray peaks are approaching the measured data. The model is capturing the intensity and accuracy of the K x-ray peaks. . . . . 39
6.5 XRF pulse height spectrum of 107.1 $^{g}/_{L}$ uranium solution and residuals. K x-ray peaks in the regions of interest are accurately captured. 40
6.6 XRF pulse height spectrum of 160.8 $^{g}/_{L}$ uranium solution and residuals. Differences from the measured data remain below 80 keV but the K x-ray emissions are accurately modeled. . . . . 41
6.7 XRF pulse height spectrum of 214.5 $^{g}/_{L}$ uranium solution and residuals. As in the 160.8 $^{g}/_{L}$ case, differences from the measured data remain below 80 keV but the K x-ray emissions are accurately modeled. . . . . 42
6.8 XRF pulse height spectrum of 268.4 $^{g}/_{L}$ uranium solution and residuals. The model is approaching the measured data below 80 keV but is more accurate in the regions of interest. . . . . 43
6.9 XRF pulse height spectrum of 323.7 $^{g}/_{L}$ uranium solution and residuals. The model is capturing the K x-ray emissions and is approaching the measured data below 80 keV. . . . . 44

6.10 XRF pulse height spectrum of 107.5 $^{g}/_{L}$ uranium solution and plutonium at 103:1 mass ratio of uranium to plutonium. As in the uranium cases the model begins to deviate from the measured data below 80 keV but dues accurately capture the uranium K x-ray emissions. The plutonium $K_{\alpha 1}$ peak is captured however the $K_{\alpha 2}$ peak is lost in the higher energy shoulder of the uranium $K_{\alpha 1}$. The plutonium $K_{\beta}$ peaks are present and relatively well represented. . . . 45

6.11 XRF pulse height spectrum of 160.8 $^{g}/_{L}$ uranium solution and plutonium at 103:1 mass ratio of uranium to plutonium. Deviation of the model from measured data remains relatively constant in energy regions of interest. Lower energies are more accurately represented between 42 and 59 keV. . . . . 46

6.12 XRF pulse height spectrum of 250 $^{g}/_{L}$ uranium and plutonium solution and residuals. Deviation of the model from measured data remains relatively constant in energy regions of interest. Lower energies are more accurately represented between 42 and 59 keV. . . . . 46

6.13 $K_{\alpha 1}$ peak intensities as a function of uranium concentration. Below 50 keV the model matches the measured peak intensity relatively well, however a significant deviation occurs at higher energies. . . . . 47

6.14 $K_{\alpha 2}$ peak intensities as a function of uranium concentration. Again, a significant deviation occurs at higher energies, while lower energies more closely match the measured data. . . . . 48

6.15 $K_{\beta 13}$ peak intensities as a function of uranium concentration. The same deviation is seen above 50 keV. . . . . 48

6.16 $K_{\beta 24}$ peak intensities as a function of uranium concentration. As in the other K x-ray emissions, a significant deviation occurs above 50 keV. 49

6.17 $K_{\alpha 1}$ peak intensities as a function of plutonium concentration. MCNP modeled response increases in magnitude as concentration of plutonium increases. . . . . 50

6.18 $K_{\beta 13}$ peak intensities as a function of plutonium concentration. As in the other aqueous solutions, the MCNP modeled peak intensity deviation from the measured data increases with concentration. . . . 50
6.19 $K_{\beta 24}$ peak intensities as a function of plutonium concentration. Lower concentration peak intensities show a consistent offset, while the higher concentration case shows a substantial increase in the deviation. . . . 51
6.20 Differences between the measured and modeled K x-ray peak intensities as a function of uranium concentration in solution. . . . . 52
6.21 Differences between the measured and modeled K x-ray peak intensities as a function of plutonium concentration in solution. . . . . 53
6.22 Pulse height spectrum of a K-edge transmission measurement of a 1.072 $^{g}/_{L}$ uranium sample. The uranium K-edge is not visible at this concentration. . . . . 55
6.23 Pulse height spectrum of a K-edge transmission measurement of a 5.459 $^{g}/_{L}$ uranium sample. The uranium K-edge not visible at 115.6 keV. . 56
6.24 Pulse height spectrum of a K-edge transmission measurement of a 15.895 $^{g}/_{L}$ uranium sample. The uranium K-edge is slightly visible at 115.6 keV. . . . . 57
6.25 Pulse height spectrum of a K-edge transmission measurement of a 48.12 $^{g}/_{L}$ uranium sample. The uranium K-edge is visible at 115.6 keV. . . 57
6.26 Pulse height spectrum of a K-edge transmission measurement of 1K-edge07.1 $^{g}/_{L}$ uranium. Model deviation from measured data is slightly greater at the uranium K-edge of 115.6 keV. . . . . 58
6.27 Pulse height spectrum of a K-edge transmission measurement of 160.8 $^{g}/_{L}$ uranium. Model deviation from measured data is slightly greater at the uranium K-edge of 115.6 keV. . . . . 59
6.28 Pulse height spectrum of a K-edge transmission measurement of 214.5 $^{g}/_{L}$ uranium. Model deviation from measured data is slightly greater at the uranium K-edge of 115.6 keV. . . . . 60

6.29 Pulse height spectrum of a K-edge transmission measurement of 268.4 $^{g}/_{L}$ uranium. Model deviation from measured data is slightly greater at the uranium K-edge of 115.6 keV. . . . . 60
6.30 Pulse height spectrum of a K-edge transmission measurement of 323.7 $^{g}/_{L}$ uranium. Model deviation from measured data is slightly greater at the uranium K-edge of 115.6 keV. . . . . 61
6.31 Pulse height spectrum of 107.5 $^{g}/_{L}$ uranium and plutonium at a mass ratio of 103:1 of uranium to plutonium. Deviation of the model from measured data remains relatively constant in energy regions of interest. Lower energies are more accurately represented between 42 and 59 keV. 62
6.32 Pulse height spectrum of 160.8 $^{g}/_{L}$ uranium and plutonium at a mass ratio of 103:1 of uranium to plutonium. Deviation of the model from measured data remains relatively constant in energy regions of interest. Lower energies are more accurately represented between 42 and 59 keV. 62
6.33 Pulse height spectrum of 243.3 $^{g}/_{L}$ uranium and plutonium at a mass ratio of 82.81:1 of uranium to plutonium. Deviation of the model from measured data remains relatively constant in energy regions of interest. Lower energies are more accurately represented between 42 and 59 keV. 63
6.34 K-edge ratio drop as a function of energy for uranium only samples. . 64
6.35 K-edge ratio drop as a function of energy for uranium and plutonium samples. . . . . 65
6.36 Difference in the modeled K-edge tansmission from from the measured data. . . . . 66
6.37 MCNP geometry of the K-edge check simulation. The source is placed at (0,-1,0) directly below the metal foil, and germanium detector. . . 67
6.38 Fractional transmission of a flat x-ray spectrum as a function of the fractional density of rhodium. At lower fracitonal densities the MCNP model over estimates the transmission of x-rays. . . . . 68

6.39 Fractional transmission of a flat x-ray spectrum as a function of the fractional density of iridium. A consistant difference between the modeled and calculated transmission was found at all fractional densities. 69
6.40 Fractional transmission of a flat x-ray spectrum as a function of the fractional density of uranium. The MCNP simulation increases in deviation from the calculated data as the fractional density of uranium increases. . . . . 70

7.1 XRF pulse height spectrum of salt taken from the INL Mk.4 electrorefiner. This electrorefiner processed blanket fuel from EBR-II. Several prominent peaks of uranium, plutonium, and neptunium are visible. . 72
7.2 XRF pulse height spectrum of salt taken from the INL Mk.5 electrorefiner. This electrorefiner processed blanket fuel from EBR-II. Several prominent peaks of uranium, plutonium, and neptunium are visible. . 73
7.3 KED pulse height spectrum of salt taken from the INL Mk.4 electrorefiner. Two distinct K-edge drops can be seen between 115 and 122 keV. . . . . 74
7.4 KED pulse height spectrum of salt taken from the INL Mk.5 electrorefiner. This electrorefiner processed blanket fuel from EBR-II. Uranium and plutonium K-edge drops are prominent between 115 and 121 keV. 75
7.5 XRF pulse height spectrum of a voloxidation powder sample. The average sample density is 16.81 $^{g}/_{cm^3}$. . . . . 77
7.6 XRF pulse height spectrum of a liquid cathode containing cadmium and plutonium 0.82 cm thick with a density of 10.97 $^{g}/_{cm^3}$. . . . . 78
7.7 XRF pulse height spectrum of a liquid cathode containing bismuth and plutonium 0.19 cm thick . Plutonium K x-ray emissions are shown in green. . . . . 79

7.8 XRF pulse height spectrum of a metallic foil of uranium 0.24 cm thick with a density of 19.05 $^{g}/_{cm^3}$. X-ray emissions of uranium fall within the grey ROIs. . . . . 80
7.9 KED pulse height spectrum of a voloxidation powder sample 0.27 cm thick. A rough uranium (the primary component of the sample) K-edge is visible at 115.7 keV. . . . . 81
7.10 KED pulse height spectrum of a liquid cadmium cathode 0.82 cm thick. 82
7.12 KED pulse height spectrum of a metallic foil of uranium 0.24 cm thick. The area around the uranium K-edge is denoted in grey. . . . . 82
7.11 Liquid cathode containing bismuth and plutonium 0.19 cm thick. The green ROI brackets the plutonium K-edge at 121.8 keV. . . . . 83
7.13 XRF pulse height spectrum of 107.5 g/L U at 103:1 U:Th mass ratio. Thorium x-ray emissions are denoted by the green ROI and uranium in grey. . . . . 84
7.14 XRF pulse height spectrum of 160.8 g/L U at 103:1 U:Th mass ratio. Thorium x-ray emissions are denoted by the green ROI and uranium in grey. . . . . 85
7.15 XRF pulse height spectrum of 213.0 g/L U at 102.4:1 U:Th mass ratio. Thorium x-ray emissions are denoted by the green ROI and uranium in grey. . . . . 85
7.16 XRF pulse height spectrum of 243.3 g/L U at 82.81:1 U:Th mass ratio. Thorium x-ray emissions are denoted by the green ROI and uranium in grey. . . . . 86
7.17 XRF pulse height spectrum of 243.3 g/L U at 100:1 U:Th mass ratio. Thorium x-ray emissions are denoted by the green ROI and uranium in grey. . . . . 87
7.18 XRF pulse height spectrum of 243.3 g/L U at 20:1 U:Th mass ratio. Thorium x-ray emissions are denoted by the green ROI and uranium in grey. . . . . 87

7.19 XRF pulse height spectrum of 243.3 g/L U at 4:1 U:Th mass ratio. Thorium x-ray emissions are denoted by the green ROI and uranium in grey. . . . . 88
7.20 XRF pulse height spectrum of 243.3 g/L U at 2:1 U:Th mass ratio. Thorium x-ray emissions are denoted by the green ROI and uranium in grey. . . . . 88
7.21 XRF pulse height spectrum of 243.3 g/L U at 1:1 U:Th mass ratio. Thorium x-ray emissions are denoted by the green ROI and uranium in grey. . . . . 89
7.22 KED pulse height spectrum of 107.5 g/L U at 103:1 U:Th mass ratio. 90
7.23 KED pulse height spectrum of 160.8 g/L U at 103:1 U:Th mass ratio. 90
7.24 KED pulse height spectrum of 213.0 g/L U at 102.4:1 U:Th mass ratio. 91
7.25 KED pulse height spectrum of 243.3 g/L U at 82.81:1 U:Th mass ratio. 91
7.26 KED pulse height spectrum of 243.3 g/L U at 100:1 U:Th mass ratio. 92
7.27 KED pulse height spectrum of 243.3 g/L U at 20:1 U:Th mass ratio. . 92
7.28 KED pulse height spectrum of 243.3 g/L U at 4:1 U:Th mass ratio. . 93
7.29 KED pulse height spectrum of 243.3 g/L U at 2:1 U:Th mass ratio. . 93
7.30 KED pulse height spectrum of 243.3 g/L U at 1:1 U:Th mass ratio. . 94
7.31 Fission product source term subtracted XRF pulse height spectrum from 45 $^{GWd}/_{T}$ spent fuel that has cooled for 5 years. . . . . 96

# Nomenclature

| | |
|---|---|
| DA | Destructive assay |
| ER | Electrorefiner |
| HKED | Hybrid k-edge densitometry |
| ICP-MS | Inductively coupled plasma mass-spectroscopy |
| ID-MS | Isotope dilution mass-spectroscopy |
| INL | Idaho National Laboratory |
| KED | K-edge densitometry |
| LBC | Liquid bismuth cathode |
| LCC | Liquid cadmium cathode |
| LIBS | Laser-induced breakdown spectroscopy |
| NDA | Non-destructive assay |
| ORNL | Oak Ridge National Laboratory |
| PUREX | Plutonium and Uranium Recovery by Extraction |
| REDC | Radiochemical Engineering Development Center |
| TRU | Transuranic |
| XRF | X-ray fluorescence |

# Chapter 1

# Introduction

Pyrochemical conditioning (pyroprocessing) of spent nuclear fuel was developed as a method to partition wastes and recover useful major actinides from metallic fuel developed for the Integral Fast Reactor EBR-II. Development of a separate process from PUREX or UREX (plutonium and uranium extraction methods) aqueous based recovery techniques was necessary as the fuel used in EBR-II was sodium bonded as a result of the liquid sodium used in the primary coolant circuit of the reactor. A unique characteristic of pyroprocessing is that is it more robust process with regards to radiation tolerance due to the absence of organic solvents used in the liquid-liquid aqueous extraction processes. This presents an opportunity to recover actinides and partition waste from fuel more recently removed from use in a reactor. This process also does not partition plutonium from other minor actinides which increases the difficulty of using this reprocessing method for proliferation (5).

While pyroprocessing shows promise as a method of recovering valuable fissile material from more recently discharged fuel from a reactor, comprehensive safeguards measurement methods must be developed. Pyroprocessing requires extremely high temperatures (greater than 800 °C) and an inert atmosphere for the process to function. The presence of a very high radiation field from the fission products present

result in a very difficult sampling process as current methods of determining the elemental composition of samples taken from a pyroprocess are different forms of mass spectroscopy. Hybrid K-edge Densitometry (HKED) is a promising candidate measurement method for pyroprocessing as it can make measurements through optically thick materials such as process barriers and denser samples from a pyroprocess and in the presence of high radiation fields.

Hybrid K-edge Densitometry (HKED) is a method currently used in commercial aqueous reprocessing operations to determine the uranium concentration in solution. HKED uses tuned x-ray beams to bombard a sample of an unknown concentration of uranium. Information from the transmission difference through a sample and its characteristic x-ray emission indicate the uranium concentration. However, scoping studies are not feasible as the instruments are rare, expensive, and not available for modification. The purpose of this research was to develop a high-fidelity computational model of an HKED instrument that was validated against measurements made with the actual instrument. Upon validation the model was used to predict the system response to a variety of samples throughout a pyroprocessing operation. These samples will come in several physical forms such as molten salt, liquid metal cathodes, powders, and solid metallic foils. Other sample characteristics such as the source terms presented by self-induced x-ray fluorescence and gamma-ray emission from active samples were investigated. Finally, simulations to determine the feasibility of using of thorium as a plutonium surrogate in a mixed actinide solution were performed. This will facilitate the creation of samples at the University of Tennessee's Radiochemistry Center for Excellence for measurement at Oak Ridge National Laboratory.

# Chapter 2

# Pyrochemical Reprocessing of Spent Nuclear Fuel

## 2.1 Overview

Pyrochemical processing, also known as pyroprocessing, is an electrochemical method of treating spent nuclear fuel, metal as well as oxide, to recover actinides from fission products for further use in nuclear fuel. A potential application of pyroprocessing includes closing the nuclear fuel cycle while providing a means of reducing the quantity of waste at commercial power plants. Partitioning the waste into its constituents enables recycling of fissile material back into the fuel cycle while possibly reducing the volume of process waste. The purpose of this literature review is to assess the current state of pyrochemical reprocessing and its unique safeguards challenges and thus glean some understanding of its future direction.

Currently, several nations including the U.S., South Korea, Japan, France, Germany, the United Kingdom, and Russia are investigating pyroprocessing technologies for their respective fuel cycles. In several countries there is growing support for, and work progressing in, deployment of alternative spent-fuel reprocessing

operations. Several countries are currently assessing pyroprocessing as a complement to aqueous reprocessing methods because of its radiation robustness, a denser more compact waste form, and its ability to recover valuable fissile material without chemically separating plutonium from other minor actinides (6). The majority of work in developing the technology required for commercialization has focused on the increasing throughput, development of a continuous process, methods for recovery of molten salt, and salt waste disposition. Most efforts are currently at the prototype scale with several proposals for full-scale facilities being fielded (7; 8).

## 2.2 Process Description

This method of reprocessing relies on the differences in the Gibbs free energies of the metallic fuel and fission products to partition the spent fuel. This is the energy available in a system to do work and since components of spent fuel have negative Gibbs free energies a net energy input is required to drive the reaction. Energy is added to the system in the form of a voltage potential across the electrolyte solution to reduce uranium out of the salt to the cathode. For example, certain fission product constituents within the fuel (e.g., isotopes of Cs, K, Sr, etc.) possess Gibbs free energies ranging from -87.8 to -65.1 kcal/mol; as a result, these species are readily absorbed by the medium and remain relatively stable in in the salt phase (9). Adding energy to the system in the form of a voltage potential applied across the electrolyte results in desirable products, in this case uranium metal, to plate to a cathode for further processing.

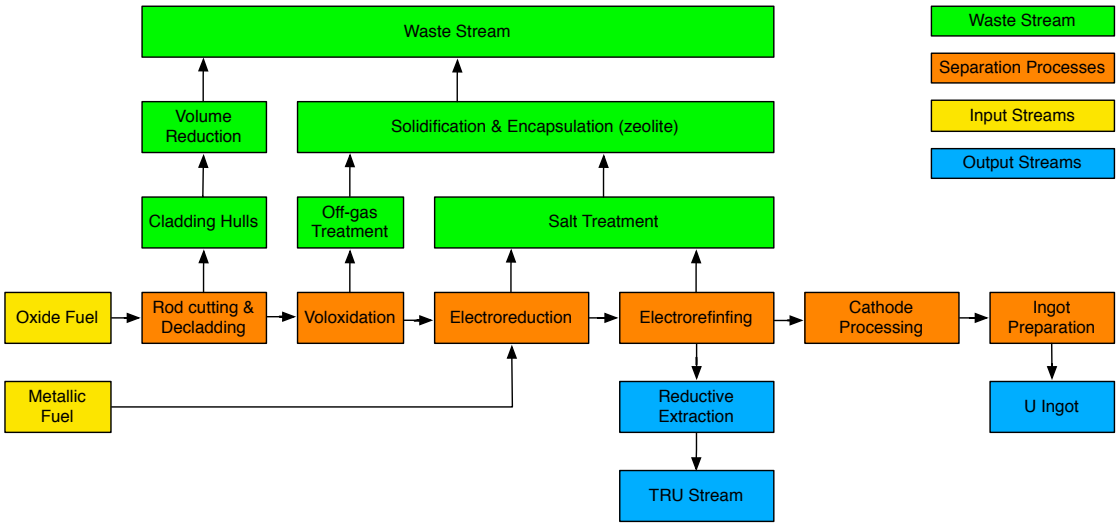


**Figure 2.1:** A pyrochemical process flowsheet for processing ceramic power reactor fuel.

Before introduction into the process line, oxide-based fuel must first be removed from its cladding using mechanical methods. Fuel assemblies are dismantled by removing both end plates and each spacer grid and individual fuel pins are then reduced in length by a mechanical chopper until the pin segments can fit into a device that cuts the cladding axially. Metal waste and any fission gases are directed to their respective waste streams, which will be discussed further in this paper.

After the cladding has been mechanically removed the individual pellets are directed to a voloxidation process where $UO_2$ is converted to $U_3O_8$ decreasing material density while increasing its volume. This process has the effect of increasing the rate of reduction as $U_3O_8$ is converted to metallic uranium. The majority of fission product gases and some of the metal elements present that are transformed to volatile oxides are removed. The resulting powder material is then placed in an electrorefiner along with some electrolytic transfer medium for conversion to a metal form. Cathodes containing metallic uranium, minor actinides and lanthanides, and any metallic fission products are then directed to another electrorefiner for further processing. At this

time there are no non-destructive assay methods for characterizing the uranium input from the voloxidation process to the electrorefiner.

### 2.2.1 Electrorefining

A key piece of equipment in pyroprocessing is the electrorefiner that contains the electrolytic medium, which allows uranium metal to be partitioned and collected. These devices, while there are some variations, are generally comprised of similar components. The electrorefiners are heated and insulated vessels that can contain the molten salt medium, spent fuel components, anode and cathode assemblies and some sort of agitation mechanism that ensures a uniform distribution of the contents of the vessel. In particular, the Mk.4 electrorefiner developed at Idaho National Laboratory contains rotating anode/cathode assembly and a cadmium pool to collect plutonium along with other minor actinides(10). While the cadmium cathode should be of concern with regards to safeguarding the material, specifically plutonium, there are no non-destructive methods to determine the plutonium content of the cadmium cathode.

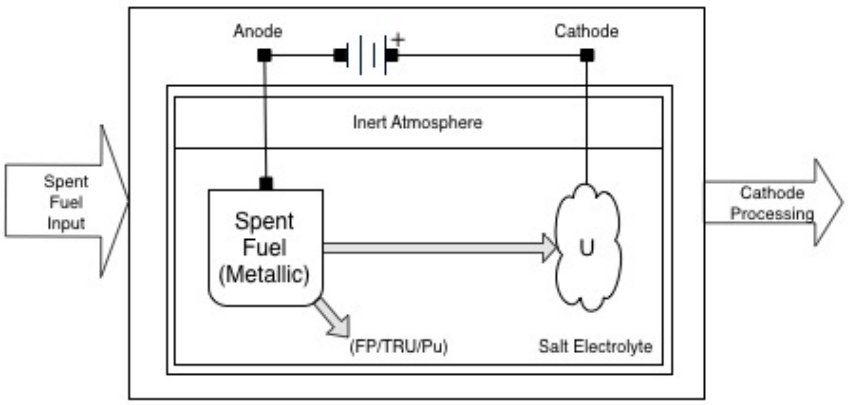


**Figure 2.2:** Material flow within a generic electrorefiner without a liquid cadmium cathode.

Figure 2.2 shows the flow of materials in a generic electrorefiner as uranium metal is electrolytically transported from the anode basket and is dendritically collected

at the cathode by applying a voltage potential across the whole system. After the process has run to equilibrium the cathode and the deposited uranium are removed from the electrorefiner and sent to cathode processing for uranium recovery. The fission products and transuranic constituents are left in the electrorefiner.

Fission products along with any transuranic species (including plutonium) are deposited into the molten salt until accumulation requires their removal. A liquid cadmium cathode may be employed in addition to a solid cathode for an additional method of uranium, plutonium, and transuranic extraction (11). In all of these stages and whether or not a liquid cadmium cathode is used the plutonium present in the spent fuel is never separate from the transuranic elements. The liquid cathode can also serve as an indicator of the amount of uranium dissolved in the molten cadmium by comparing the voltages of the reference electrode to the cadmium electrode (12). The molten cadmium can also be used as a second electrode by catching dendrites that fall from the cathode and dissolving them in the cadmium pool for later recovery.

The uranium, transuranics, and rare earths metals are dissolved into a LiCl-KCl eutectic salt from the anode basket. Within the electrorefiner, the uranium grows dendritically on the cathode. The uranium deposits are then processed further to cast solid uranium ingots. Rare earth elements, fission products, and transuranics chlorides remain in the salt eutectic because of their thermodynamic affinity to remain in a metal chloride state (13). This eutectic salt can be sent to an electrowinning process that will remove the remaining U and TRU constituents, as well as oxidize a portion of the rare earth constituents.

After the desired amount of uranium has been deposited on the cathode it is removed and directed to cathode processing and uranium ingot production. Metallic uranium deposits as dendrites on the cathode along with some entrained salt. The entrained salt is removed via vacuum evaporation in a screw kiln salt evaporator

between 700 °C and 900 °C and 0.2 to 0.5 torr then recondensed (14). Uranium dendrites are then directed to an induction furnace where it is melted at 1300 °C and poured into a graphite crucible to form a product metal ingot.

### 2.2.2 Salt Regeneration and Impurity Occlusion

As the fuel is processed, fission products and actinides accumulate in the salt. These impurities depress the freezing point and electrochemical properties of the melt and have a deleterious effect on the separation process. To avoid a decrease in efficiency, fresh salt must be added (which would be expensive and generate a large amount of HLW). Alternatively, the electrolyte can be regenerated; resulting in fresh salt for the separation while isolating the actinides and fission products. Waste salt regeneration begins after the actinides have been removed through an electrowinning system. Several groups across the world are working on regeneration methods, with the US, Russia, and Korea leading the way (10; 15). This diversity in study has lead to several different approaches, each with technical challenges that must be addressed. The remainder of this section will discuss some of the most recent approaches; along with a brief discussion on the effects salt regeneration may have on the safeguards design of a facility.

To date, the majority of salt regeneration work has been focused on forming insoluble precipitates of fission products or selective occlusion of fission products in an adsorbent medium. Due to the technical challenges of adsorption, many groups have focused on precipitation. In these techniques, the impurities are precipitated after introduction of a chemical species such as a phosphate or carbonate (16; 6). One method that has been and is continuing to be investigated by researchers at KAERI (Korean Atomic Energy Research Institute) is the carbonation (strontium removal) and oxidation (rare earth removal) of waste salt (16). To remove the strontium from the waste salt, $Li_2CO_3$ is added to the molten electrolyte and agitated for

approximately four hours. After cooling, the salt was dissolved in water. The insoluble $SrCO_3$ precipitates and can be separated by filtration. The reported conversion efficiency of this process ranged from 45.5 to 99.5%, depending on $Li_2CO_3/SrCl_2$ molar ratio (16). The rare earth elements were removed from the waste salt through sparging oxygen gas into the molten electrolyte. As the oxygen moves through the melt, rare earth oxychlorides precipitate and accumulate at the bottom of the melt (16). After cooling, the precipitate rich layer at the bottom of the melt can be separated and further processed, while the remaining salt is pure enough for reuse. The remaining salt in the precipitate layer can be removed by vaporizing the salt at high temperature (approximately 1200 °C) in a vacuum. This process enables almost all of the salt to be regenerated, and minimizes the waste volume of the rare earth oxides (16).

There has also been interest in removing some fission products such as strontium and caesium through crystallization techniques. This would reduce the use of additive chemicals during the regeneration step, as well as avoid the technical challenges in using chemical or ion exchange interactions. The crystallization techniques have a variety of experimental setups, but all exploit the segregation of impurities to the melt phase as the molten salt cools. This method will be utilized in the planned demonstration facility in Korea for Sr and Cs removal (15).

In the US, the majority of the salt waste research has involved the sorption of the fission products by zeolites (10). The details of this method will be discussed further within the waste disposition section of this paper, but a general description is necessary to understand the safeguard implications. Zeolite is a common commercial adsorbent that occludes ions within the pores of its structure. Both the electrolyte and the impurities are occluded in this method, resulting in an increase in waste and demand for fresh salt to be added to the system. For this reason, zeolite loading has been considered for use in conjunction with alternative salt purification methods to

reduce waste volume.

As stated earlier, the nuclear material has been removed before the start of the salt regeneration step. However, there are a number of fission products of interest (among which are Cs and Sr isotopes). These interests vary from manipulating waste management to utilizing nuclides as radiation sources. The fission products separated from the electrolyte must be accounted for and disposed of properly. These products are process dependent and can be phosphates, carbonates, or oxides. Although international safeguards do not require the safeguarding of these materials, continuous or routine monitoring of the waste products should be adopted to verify that source material has been removed prior to regeneration. It is necessary to ensure this pathway is not used for diversion as the salt once contained uranium and other minor actinides. An assay of the salt and liquid cathode material (if applicable) is required to verify that any uranium, plutonium, or other minor actinides are accounted for.

### 2.2.3 Final Waste Form

The ceramic waste form produced from pyroprocessing increases the volume of waste through the additional salt, zeolite, and glass. Thus the reduction of salt waste volume is of high importance. Methods to reduce the final salt waste include separating out lanthanides via cooxidative precipitation, noble metal separation based on placement of anode and cathode, separation of minor actinides, earth elements, alkaline earth elements, and alkali metals through an aqueous/acid/organic separation, CsCl and $SrCl_2$ separation by crystallization (17; 18; 19; 20). The production of the ceramic waste form has varied slightly over time with one significant alteration in the final processing step. The transuranic metals and most of the fission products are bound as chlorides in the salt mixture and ground in an inert argon atmosphere then mixed with ground zeolite(21; 22).

The zeolite is dried in a heated retort under vacuum to reduce moisture therefore preventing water from interfering with the salt-zeolite mixing and final waste forms ability to retain waste. Though the ideal degree of drying is unknown, a safe limit is less than 1 wt.% water. After mixing, heating, and cooling of the salt/zeolite mixture, a V-shaped blender is preferable for industrial scale, a borosilicate glass frit is added in and moved to the final step of the process. A variety of glass mixtures have been tested and it is unclear what the optimal compositions would be and if it would be preferable to use a preexisting commercial blend or create a custom blend with slightly better properties. The two methods of creating the final waste form once the glass frit is mixed in are hot isostatic pressing (HIP) and pressureless consolidation processing. HIP has been the standard method for final processing though pressureless consolidation has begun to be used and provides certain benefits such as releasing all moisture in the final phase and thus having a reduced impact of moisture present in the zeolite.

# Chapter 3

# Reprocessing Safeguards

## 3.1 Aqueous Reprocessing

Safeguards material accountancy measurements, as applied to reprocessing operations, focuses on quantifying the mass of and tracking special nuclear materials as they travel through a a reprocessing operation. This process begins as soon as material enters a process and continues as material move from one Mass Balance Area (MBA) to another. Determining where to take samples for assay requires understanding the nature of aqueous spent nuclear fuel reprocessing. The primary process involved in reprocessing is the PUREX (Plutonium and Uranium Recovery by Extraction). This process is outlined by diagram in Figure 3.1.

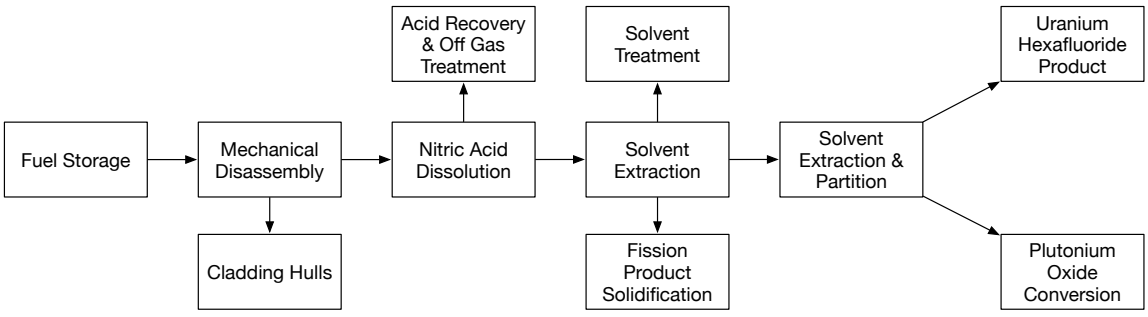


**Figure 3.1:** Flowsheet illustrating the flow of material in PUREX reprocessing (2).

The PUREX process has been employed both in the nuclear weapons complex and the civilian fuel cycle since the Manhattan Project. It is a continuous extraction process that relies on the different oxidation states of uranium and plutonium to selectively extract and partition both actinides from fission product containing waste. Spent fuel enters this process in the form of intact fuel assemblies discharged from a reactor. The assemblies must have cooled enough to facilitate handling in a hot cell without risk of rapid oxidation of the zircalloy cladding materials. The spacer grids and end caps are removed freeing the individual fuel pins for further disassembly. These pins are then chopped into shorter sections and placed in a nitric acid bath to dissolve the fuel meat but not the zircalloy cladding hulls. The cladding hulls are then discarded as high-level waste (2).

The aqueous solvent containing special nuclear material consists of uranyl nitrate ($UO_2(NO_3)_26H_2O$), plutonium nitrate ($Pu(NO_3)_4$), and various fission products. This feed is passed into a liquid-liquid extraction column with an organic solvent (tributyl-n-phosphate) (23). While separation in a single extraction column is limited, the continuous nature of aqueous reprocessing allows cascades of columns to operate in multistage countercurrent modes. Very high separation factors and process throughput are achieved by operating in this fashion. Figure 3.2 illustrates the multistage-countercurrent concept.

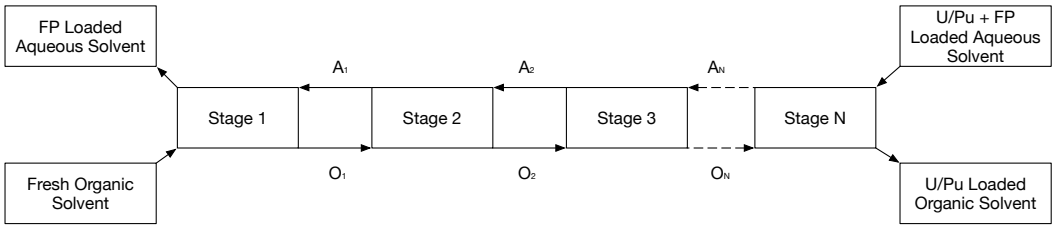


**Figure 3.2:** Countercurrent-multistage continuous flow through multiple extraction columns (2).

The actual equipment to perform the liquid-liquid extraction consists of a tall column with perforated plates distributed along the long axis. Immiscible streams enter from opposite ends and mix throughout the working section. A gas pulse is

injected into the lower section of the extraction column to further agitate the liquid on the interior and increase the contact surface area between the two phases of liquid. A generic design of a liquid-liquid extraction column is depicted in Figure 3.3.

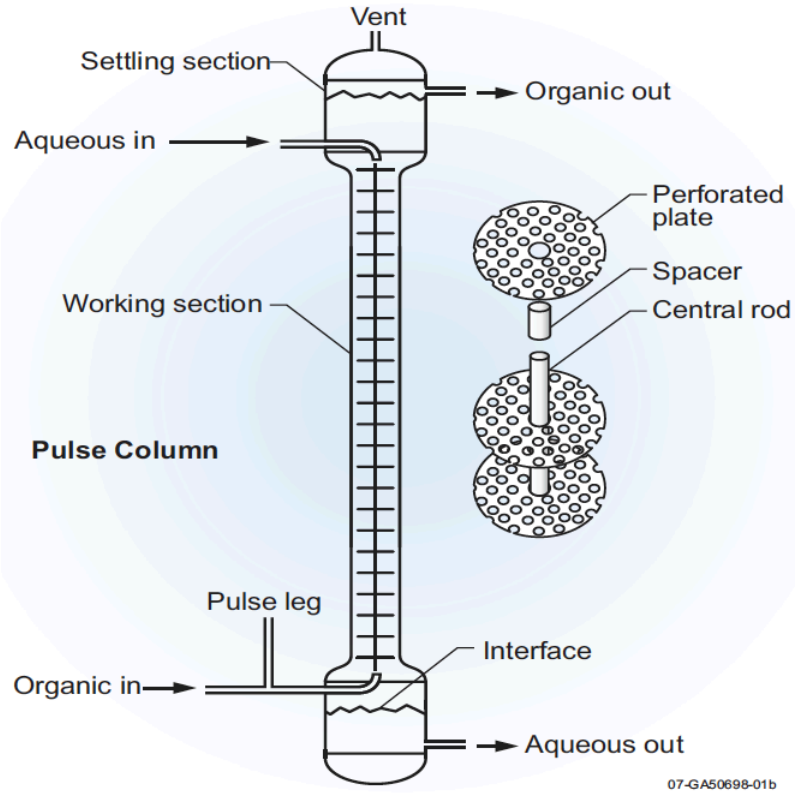


**Figure 3.3:** Typical liquid-liquid extraction column (3).

This process of liquid-liquid extraction is repeated stripping the fission products from the uranium/plutonium solution. Then uranium is partitioned from plutonium. The uranium is then converted to $UF_6$ and loaded into a cylinder for shipment to an enrichment facility for re-enrichment to a reactor usable assay. Plutonium separated from the uranium product is converted to plutonium oxide ($PO_2$) and directed to a fuel fabrication facility for mixed-oxide fuel. Solvent loaded with fission products is recovered by distillation leaving the remaining waste for disposal. The waste stream from this extraction is directed to vitrification processes in operations in both the United Kingdom, France, and Japan (3).

## 3.2 Measurement Methods

Measurements from an aqueous process are broken into two distinct types: non-destructive assay and destructive assay. The PUREX process allows for accountancy tanks before key steps in the process from which samples may be taken from these tanks and destructively assayed to quantify the uranium and plutonium concentration in solution. Samples are routinely taken as the material moves from one MBA to another ensuring that there is no diversion throughout the reprocessing operation. These types of measurements destroy the sample in the course of the measurement normally through the preparation process. Inductively coupled mass-spectroscopy (ICP-MS) or isotope-dilution mass-spectroscopy (ID-MS) are used to measure the isotopic composition in a given sample (24). While these assay methods are accurate in their determination of sample content they are time-consuming. Samples must be prepared by dilution to a measurable concentration by a specific instrument as in ICP-MS or spiked with an standard isotope and measured as in ID-MS.

Non-destructive assay techniques are able to determine the composition of a sample and return the sample to the process. These methods include laser-induced breakdown spectroscopy and hybrid K-edge densitometry. LIBS relies on measuring the visible light emitted from a plasma created by bombarding a sample with a laser. While this method cannot determine the isotopic composition of a sample it is more timely and does not require extensive sample preparation however, the LIBS method only analyzes the outermost layer of any sample(25; 26). Density measurements in reprocessing measurements are also used to determine the uranium content of a solution of known composition. Figure 3.4 illustrates the layout of a dip tube measurement apparatus.

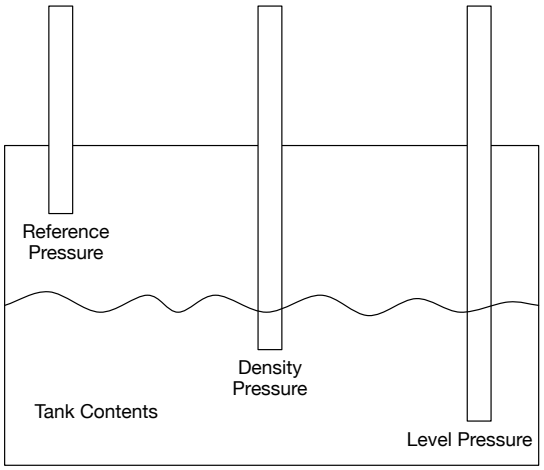


**Figure 3.4:** Typical layout of pressure sensors in a pneumatic dip tube measurement setup (4).

Several dip tubes may be mounted through the process to measure the density of a solution based on the pressure differences at each tube. Pressure measurements at each sensor allow level and density to be calculated if the tank volume and the solution temperature are known. Signals from the at the level pressure tube are compared to the signals from the density pressure tube. The height difference between the tubes is known the material level in the tank can be calculated (4; 27; 28).

Hybrid K-edge densitometry is another method to determine the elemental composition of a sample from a reprocessing operation. This method relies on quantifying the elemental density of a sample, uranium in the case of pyrochemical reprocessing, by measuring the absorption at an element's K-edge. An x-ray fluorescence measurement is combined with this measured transmission drop to determine the elemental composition. This method will be discussed in detail in the following chapter.

Each of these measurement methods, while proven in aqueous reprocessing methods, find limited applicability in pyrochemical reprocessing operations. The higher activity of samples from aqueous reprocessing methods complicate sample preparation for ICP-MS and ID-MS measurements. The multiple component nature

of pyrochemical samples complicates any measurement due to competing signatures from each constituent component. Multiple components also complicate any density measurements as a dip tube bubbler requires the density of the solution to be known. Laser-induced breakdown spectroscopy can be used to make measurements through a process barrier, however this would only measure the composition of the top layer of material in the electrorefiner. Environmental characteristics also complicated removing a sample for assay from an electrorefiner found in pyroprocessing is difficult due to the higher temperatures and the requirement of an inert atmosphere.

# Chapter 4

# Hybrid K-edge Densitometry

## 4.1 Overview

For pyroprocessing to be implemented as an effective method for recovery of valuable fissile material the development of appropriate material safeguards methods must be completed. One measurement method employed in aqueous reprocessing operations is hybrid K-edge densitometry (HKED) which is currently utilized in both AREVA's La Hague site and the Rokkasho Reprocessing Plant to determine the uranium and plutonium content of samples taken from an aqueous process. This measurement method relies on dual measurements of a sample: K-edge densitometry and x-ray fluorescence.

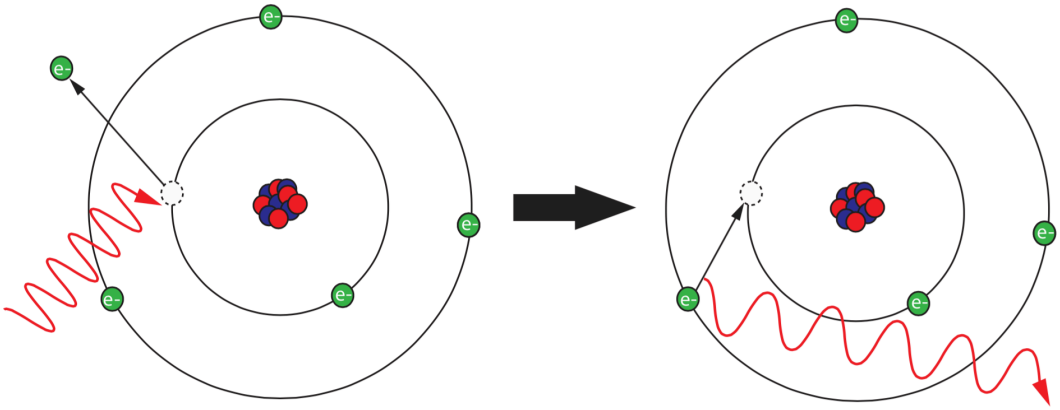

**Figure 4.1:** Physical processes behind HKED measurements.

K-edge densitometry relies on measurement of the transmission difference at an element's K-edge, which corresponds to the binding energy of the K-shell electrons. As an x-ray beam passes through a sample the photons are preferentially absorbed at this energy, resulting in a noticeable sharp drop in x-ray transmission. The increase in the attenuation coefficient that makes this transmission drop so pronounced is shown in Figure 4.2.

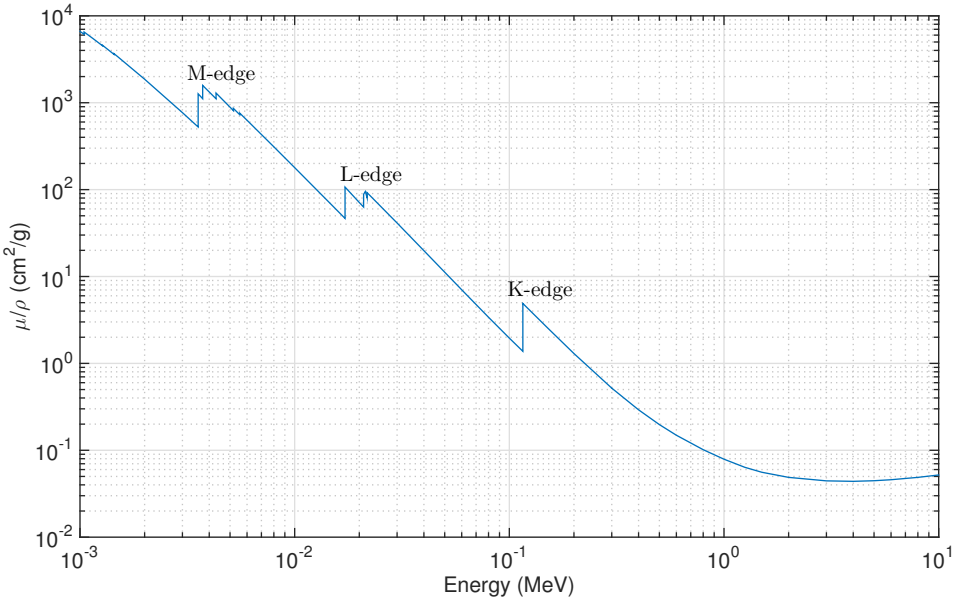


**Figure 4.2:** Photon absorption edges for uranium.

This difference is used to calculate the major actinide density in a sample, normally uranium. This measurement does not reveal the elemental composition of a sample but does indicate the total actinide concentration in solution. A vacancy is produced in the element's K-shell as a result of the x-ray absorption. When a higher shell electron de-excites to fill the vacancy a characteristic x-ray is emitted. These characteristic x-rays are produced at a variety of energies depending on which higher shell electron drops to fill the vacancy. The XRF measurements are indicative of the identity, based on the energy of the x-ray emission, and concentration, based on the intensity, of the constituent element. These peak intensities and energies are used in conjunction with the K-edge measurement determination of total actinide density reveal the identity and concentration of elements in a sample (29). Figure 4.1 illustrates this process. Table 4.1 shows the K x-ray emissions and K-edge energies for plutonium and uranium.

**Table 4.1:** X-ray emission energies and K-edges for uranium and plutonium (1).

| **Element** | **Energy (keV)** | | | | | | |
|---|---|---|---|---|---|---|---|
| | $\mathbf{K}_{\alpha 1}$ | $\mathbf{K}_{\alpha 2}$ | $\mathbf{K}_{\beta 1}$ | $\mathbf{K}_{\beta 2}$ | $\mathbf{K}_{\beta 3}$ | $\mathbf{K}_{\beta 4}$ | **K-edge** |
| **U** | 98.43 | 94.65 | 111.3 | 114.5 | 110.4 | 114.8 | 115.6 |
| **Pu** | 103.7 | 99.52 | 117.2 | 120.6 | 116.2 | 120.9 | 121.8 |

Photon beams used in HKED measurements allow this method to potentially measure samples through optically thick materials. This is an important characteristic as pyroprocessing requires an inert atmosphere and operates at high temperatures within thick-walled electrorefiners. Measurements need to be completed with samples that have 10 to 15 times the density of those from an aqueous-based process (30; 31). Measurement of multiple elements other than uranium such as plutonium, curium, americium, and neptunium can also be achieved with HKED, however larger concentrations of other minor actinides to plutonium have not been attempted (32).

Adaptation of the HKED measurement method to pyroprocessing requires the development of a computational model of the actual instrument that would allow computational prototyping without modification of an instrument. Pyroprocessing samples are more complex with regards to the number of elemental components and physical form. Issues such as over attenuation of the x-ray beam from a denser sample and competing peaks from equal concentrations of uranium and plutonium need to be explored. Scoping studies that involve modification of the instrument and/or measurement of new sample compositions would require the use of mixtures containing active fission products as well as other actinides. A validated model would provide the ability to perform scoping studies without the requirement of active material and associated support infrastructure. This is especially applicable for pyrochemical solutions as greater concentrations of fission products and actinides would require the use of a hot cell.

## 4.2 Instrument Description

The Canberra Hybrid K-edge Densitometer consists of several important components that are crucial to developing a correct model of the system (Figure 4.3).

The core of the instrument is a x-ray tube that serves as the photon source for the K-edge transmission measurement and the excitation source for the x-ray fluorescence measurement. Directly adjacent to the x-ray tube is the sample holder tube where the sample is held within a carriage. This tube serves as the splitting point from which the two measurements are made. The K-edge densitometry beamline is directly opposite on of the sample holder and x-ray tube. Each component beamline is surrounded by tungsten shielding in addition to lead shielding on the exterior of the instrument as the instrument is designed for the operator to be within close proximity to the instrument as measurements are being taken.

Several filters are in place to tune the x-ray beam and optimize each detector's response. The x-ray source itself is contains both a beryllium and aluminum filter to

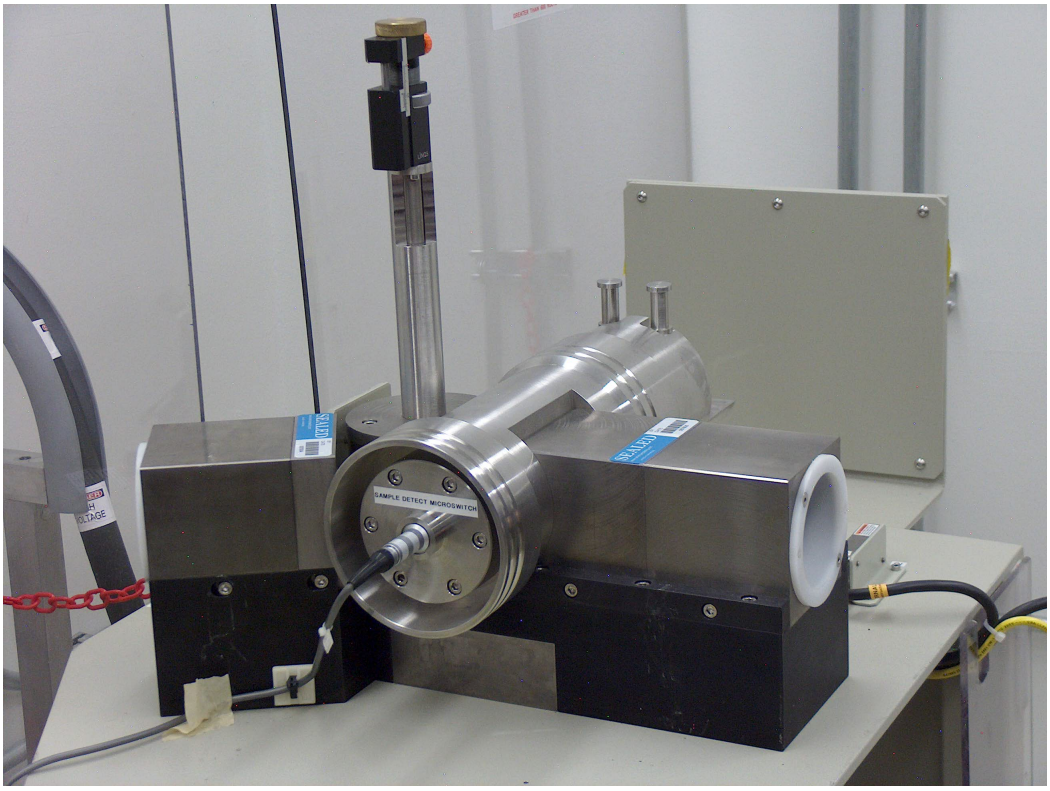


**Figure 4.3:** HKED system without outer lead shielding. Both XRF (left) and KED (right) component beamlines are exposed and germanium detectors have been removed

seal the x-ray tube and provide a window that will not greatly attenuate the beam. Directly beyond the sample position along the KED beamline is an iron filter to attenuate any lower-energy photons from the x-ray source and reduce the direct x-ray flux on the K-edge detector. This filter has the effect of decreasing the deadtime in the KED detector due to the direct photon flux from the x-ray source. A gadolinium foil is placed at the sample end of the XRF beamline to provide calibration peaks for the XRF detector and as a filter for the XRF detector. At the end of each beamline is a $^{109}$Cd source for energy and intensity calibration as the $^{109}$Cd source will remain relatively constant in activity in relation to the x-ray tube operating characteristics.

# Chapter 5

# Monte Carlo Modeling

## 5.1 HKED Model

### 5.1.1 Overview

This model was developed using the Monte Carlo N-Particle code, MCNP6.1. MCNP can effectively simulate the transport of photons throughout the system to the detectors and effectively model the detector response for a given sample and system configuration. Previous models by Berlizov and Farr of this system have been performed, however their application has been limited and required a complete redesign of the model (33; 34) . The work completed by Berlizov was based on a different model of HKED instrument and focused solely on the XRF response to a binary solution of uranium ad plutonium. Farr's modeling efforts, while adequate for a first order approximation of a model, did not incorporate the more advanced photon libraries and only modeled photon transport excluding electron physics. Inclusion of electron physics is necessary to reproduce the physical processes that are occurring in the instrument correctly. This is especially important when considering the Compton continuum.

To develop an effective model required disassembly of the system and measurements of all components that could be removed which included the shielding and

collimators for both the KED and XRF beamlines. The sample holder tube where the sample is held was also removed and measured. These dimensions were then compared to schematics to assist in building the geometry of the system in MCNP.

As the goal of developing the model is to predict the system response to a variety of samples, the system output must be accurately modeled. Producing an accurate representation of the system output was accomplished using a F8 pulse height tally in the model. This tally performs a balance of energy deposited into and lost from the tally volume which corresponds to a pulse. These pulses can be binned by energy and tabulated as a relative contribution from the number of source particles. F8 tally pulses may also be modified to include Gaussian Energy Broadening to produce realistic energy spectra that account for changes in the full width half-maximum characteristics for a given type of detector. Electron physics were also enabled in the model to more accurately capture the underlying physics in the problem.

The x-ray source for this model was developed using SpekCalc, an application used to estimate x-ray beam profiles for medical linear accelerators with a variety of tube parameters. The characteristics for the x-ray source included the beryllium filter present in the MXR-160 x-ray tube in the HKED instrument and a maximum accelerator voltage of 150 kV and the lower energy was restricted to 15 kV by the program. Figure 5.1 shows the x-ray spectrum generated by SpekCalc.

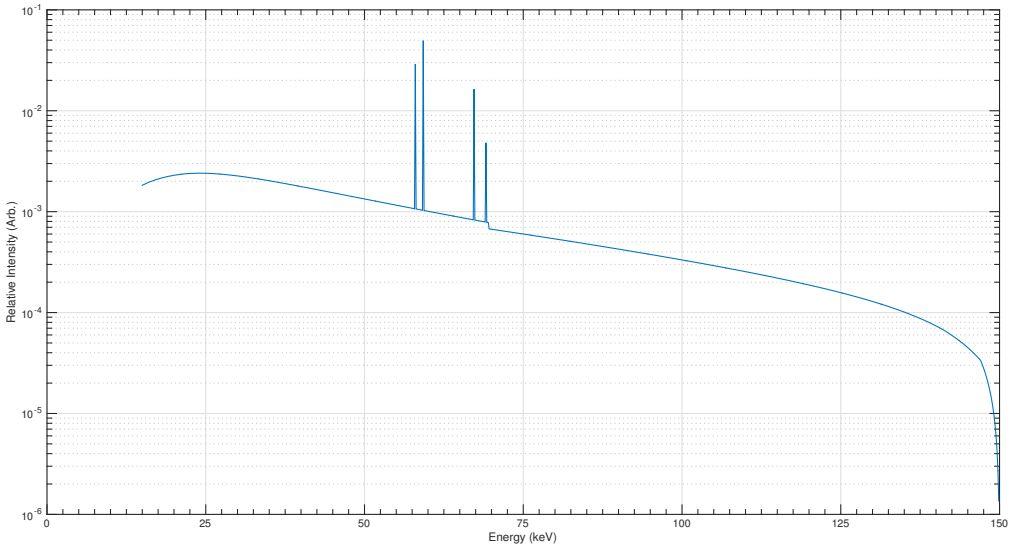


**Figure 5.1:** X-ray spectrum generated using SpekCalc. Maximum accelerator voltage was set to 150 keV.

As the x-ray source has never been well characterized for the HKED instrument and can be operated at different energies and intensities, this was chosen as a close approximation to a first principles model of the x-ray tube.

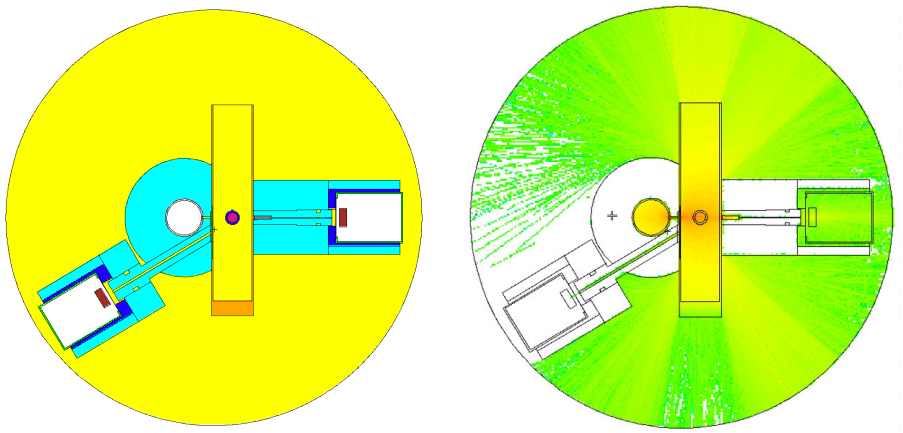

**Figure 5.2:** Left: HKED geometry generated by MCNP. Right: MCNP mesh tally showing the particle population in the system geometry. Note that the photon population is vastly greater in the K-edge beamline.

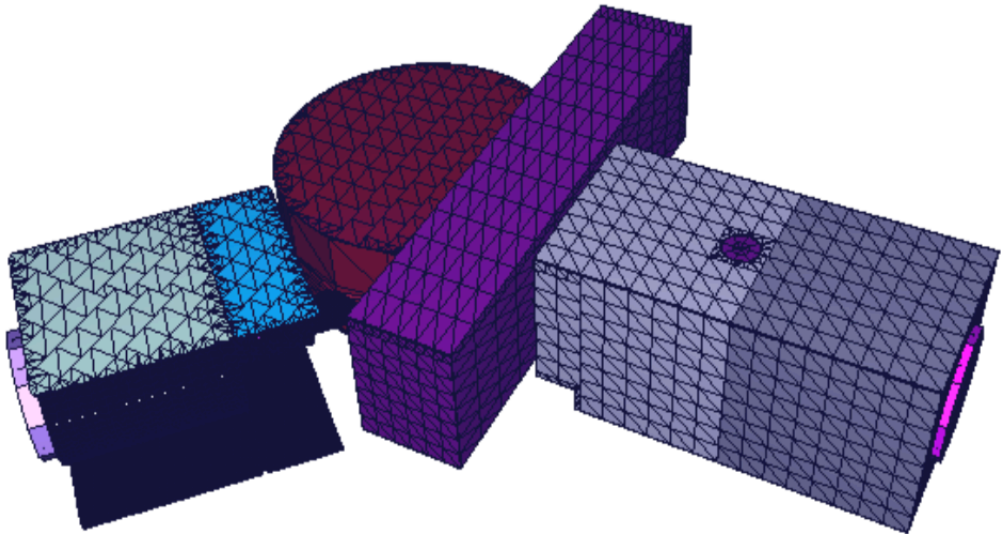

**Figure 5.3:** 3D rendering of the HKED model in VisEd. The XRF beamline is shown on the left and the KED beamline on the right.

Figures 5.2 and 5.3 show the geometry of the HKED system within MCNP as well as a mesh tally. The KED beamline and detector is located to the right side of the instrument while the XRF components and detector are located in the lower left

quadrant. The x-ray tube is located in a tungsten shield to the left of the sample holder.

The first series of simulations focused on determining the differential pulse height spectrum in each of the low-energy germanium detectors at the aperture of each beamline. Simulating the particle transport in this was proved to be very time consuming as a single case would take approximately 48 hours and $10^{12}$ particles for the case to converge with acceptable statistical variance. Monte Carlo variance reduction methods were applied to increase the computational efficiency by speeding the convergence for each tally. Since fewer particles were required for convergence the simulations were able to run to completion more quickly.

### 5.1.2 Variance Reduction Methods

As each beamline differed in its design and method of measurement, a unique variance reduction strategy was required for both XRF and KED beamlines to speed up simulation runtime and improve the statistical problem convergence. A solution to applying a unique strategy to each component was to divide the problem into two parts: a XRF component and a KED component. This strategy enabled variance reduction strategies to be tailored to a specific component of the HKED system. However, the implementation of variance reduction limited the type of tally that could be used in the problem. Since a pulse height tally (F8 tally) was desired, a solution to this was to further divide the modeling of particle transport through each component into two stages. The first stage of each simulation involved the application of variance reduction techniques to efficiently transport photons to a small disk volume located at the aperture of each beamline, termed a *microcell*. A surface flux tally (F2 tally) was used in the first stage simulations on this volume. Upon completion, post processing was performed using a Python utility to extract and reformat the tally from the first simulation to a source term used in the second stage simulation. The second stage simulation consisted of determining the pulse-height tally detector response in each

detector crystal from the source term calculated in the first stage simulation. The pulse height tally results were then normalized to a $^{109}$Cd gamma-ray peak at 88 keV for comparison to the measured data.

### XRF Variance Reduction

Modeling XRF components required a more complex method of variance reduction than the KED component due to the lower photon flux at the XRF detector. The variance reduction strategy included:

- **Forced collisions:** Photons entering sample are forced to interact
- **Deterministic transport sphere (DXTRAN sphere):** Particles preferentially scatter to sphere volume
- **Spatial weight window mesh:** Particle population control and weight distribution as a function of position
- **Energy weight window mesh:** Particle population control and weight distribution as a function of energy

Initial simulations without any variance reduction showed that the population of photons tallied in the detector volume was much lower than that required for the problem to statistically converge. Photon collisions were forced to occur within the sample material, thereby increasing the population of photons originating from a x-ray fluorescence event in the sample. These photons were then preferentially scattered to the microcell tally volume using a deterministic transport sphere (DXTRAN sphere). This sphere was placed at the far end of the XRF beamline around a thin disk at the aperture. Implementing the DXTRAN sphere was required due to the lower probability that photons would scatter into the XRF beamline.

One side effect of employing a DXTRAN sphere was an increase in the number of particles with higher-than-average weights reported in the DXTRAN diagnostics output. When particles are scored in a tally, the number of particles and their

associated weight are recorded. Particles from events with a low probability of occurrence (such as scattering through a thick media) possess higher-than-average weight. These high weight particles decrease the tally accuracy by increasing the variance since Monte Carlo calculations are based not only on the number of particles at a specific energy passing a surface but also their weight. A solution to this was to employ a spatial weight window mesh to control the splitting and removal of particles from the system. The weight window mesh controlled the severity of particle splitting as photons scattered towards the region of interest and how particles were removed from the system via Russian Roulette. In both of these processes the weight was either redistributed among the surviving particles in the case of Russian Roulette or divided amongst the new particles after splitting. An energy-based weight window mesh was also incorporated to increase the number of photons in the energy range of interest since the x-ray source, by its nature, is more intense at the lower energies. The energy range of interest corresponds to the binding energies of the K-shell electrons. Generating particles in this range decreases the overall computer time required for problem convergence because the photons generated are more likely to interact in the sample. (35).

### KED Variance Reduction

Variance reduction of the KED component of the system required a less complex approach than that of the XRF component. The following variance reduction methods were employed:

- **Deterministic transport sphere (DXTRAN sphere):** Particles preferentially scatter to sphere volume
- **Spatial weight window mesh:** Particle population control and weight distribution as a function of position
- **Energy weight window mesh:** Particle population control and weight distribution as a function of energy

As the KED beamline is in the direct line of fire of the x-ray source, a shortage of photons in the was not an issue to these simulations. Energy splitting was again employed to increase the relative population of photons in the energy range of interest between 110 and 140 keV, which corresponds the K-edges of major and minor actinides. Due to the presence of beam filter (designed to attenuate lower-energy photons), higher-weighted particles were scored in the tally microcell at the KED beamline aperture (resulting from the low probability of lower-energy photons scattering through the beam filter). A solution to this was another spatial weight window mesh that split particles as they approached the tally volume and redistributed their weight amongst the split particles. Lower energy particles are not likely to interact in the detector volume due to a beam filter. Since these low-energy photons rarely influence to the problem tally but would still require computational time, an inverse energy splitting option was used to reduce the particle population by an order of magnitude in regions not related to the problem.

The effect of this solution strategy was a decrease in the number of particles required for statistical convergence from $10^{12}$ for a combined simulation to $5 \cdot 10^7$ and $2 \cdot 10^6$ particles for stage one simulations of the XRF and KED components, respectively. Stage two simulations were completed using $5 \cdot 10^6$ particles for both XRF and KED components.

### 5.1.3 Photon Library Comparison

Other studies of the XRF response of the system revealed that the photon libraries provided with MCNP were not accurately modeling the $K_\beta$ fluorescence peaks. Previous versions of the MCNP photon libraries, specifically *mcplib84*, employed a weighted average to represent the $K_\beta$ lines as two emissions (36). This results in misrepresentation of the $K_\beta$ lines by combining the $K_{\beta_{1-5}}$ transitions into two emission lines.

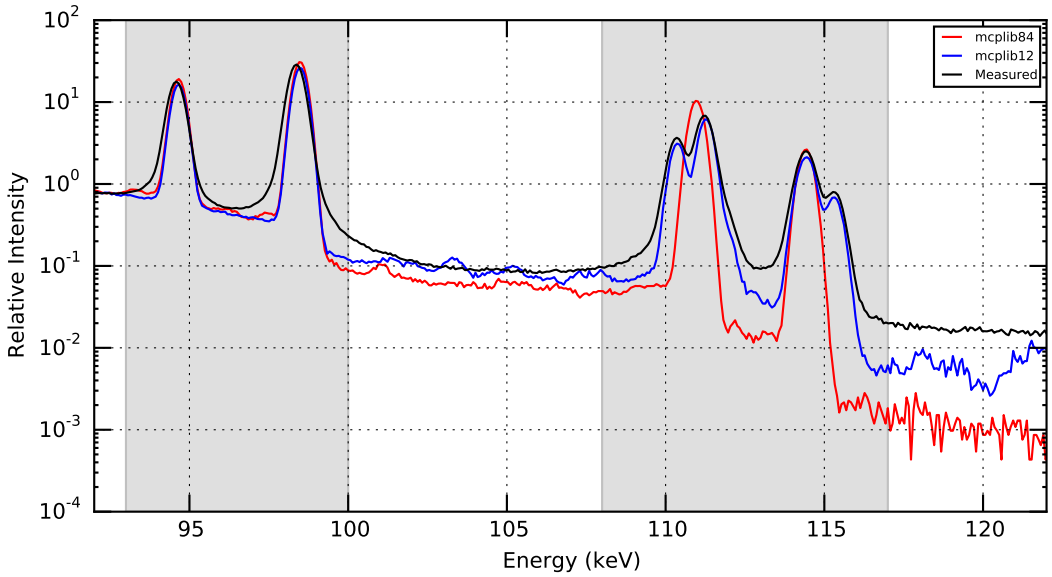


**Figure 5.4:** Pulse height spectrum of 323.7 $^{g}/_{L}$ uranium using *mcplib84* and *mcplib12*. $K_{\beta}$ peaks are not well represented at between 110 and 115 keV using *mcplib84*, however the $K_{\beta}$ peaks are correctly modeled using *mcplib12*.

Resolution of the $K_{\beta}$ peaks was required to accurately predict the system response especially with solutions containing multiple actinides. The default photon library *mcplib84* is shown in red and under represents the $K_{\beta}$ peaks. A newer photon library released with version 6.1, *mcplib12* was chosen for the model validation simulations (37). This photon library models the L-shell transistions that produce $K_{\beta}$ emission lines. The properly represented $K_{\beta}$ peaks are shown between 110 and 115 keV as indicated by the data represented in green in Figure 5.4.

Using this newer library resulted in a decrease in computational performance. Run times were increased by an average factor of two when *mcplib12* was used instead of *mcplib84* due to the more detailed physical processes modeled with the newer library. The performance penalty results from a greater number of modeled electronic transitions than previously handled in *mcplib84*. The newer library also extends to lower energies than the older library which results in a greater run time to track a greater number of lower energy particles produced, however this was mitigated by disabling the detailed physics models below 1 keV.

### 5.1.4 Pulse Height Spectrum Shift

Previous work completed on modeling of a HKED system reported an approximate 0.5 keV shift in the modeled spectrum from measured data (33). Analysis of simulations using the model developed for the scope of this work showed that the average energy offset of a simulated spectrum to measured data was approximately 0.707 ± 0.014 keV, as depicted in Figure 5.6.

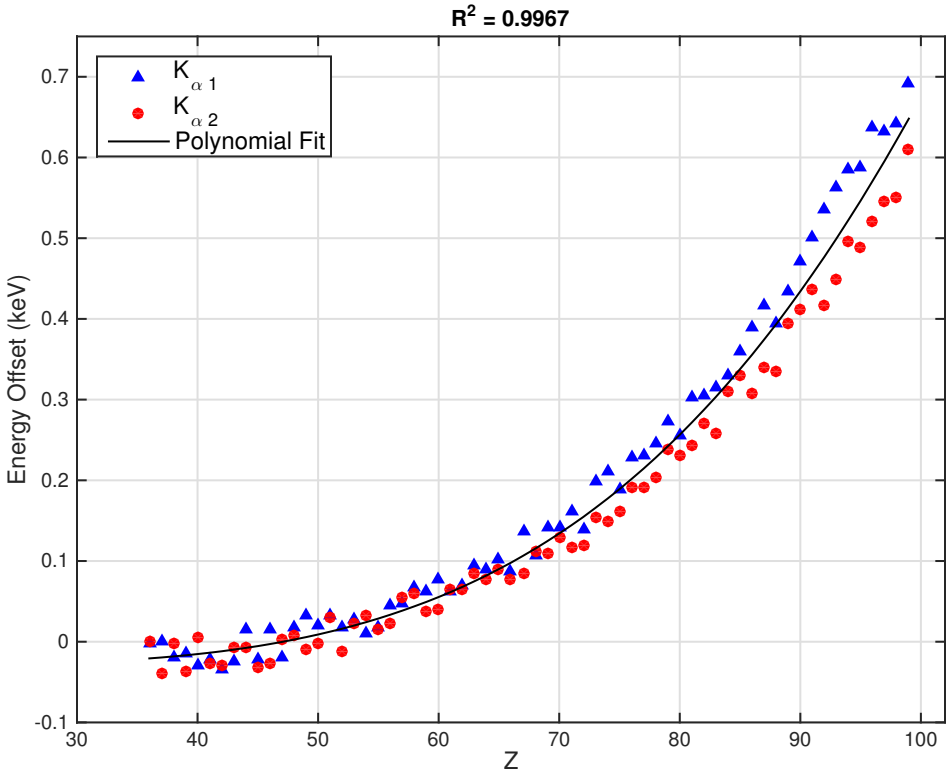


**Figure 5.5:** Calculated energy offset for both $K_{\alpha 1}$ and $K_{\alpha 2}$ photon emissions from a variety of simulations in MCNP compared to NIST XRF emission energies. The polynomial fit to these data sets is also shown with an $R^2$ value of 0.9967.

Figure 5.5 shows the calculated $K_\alpha$ energy offset from each simulation. This offset was calculated by comparing the MCNP generated x-ray emission to energies found in the National Institute of Standards and Technology X-ray Transition Energies database (1). The difference between the MCNP generated $K_{\alpha 1}$ and $K_{\alpha 2}$ emissions

and the NIST reported emissions vary separately. Using Matlab's Curve Fitting Toolbox a third order polynomial was produced to fit both $K_\alpha$ and $K_\beta$ datasets and follows the form:

$$f(x) = p_1 * x^3 + p_2 * x^2 + p_3 * x + p_4 \tag{5.1}$$

**Table 5.1:** Parameters and confidence bounds for the polynomial fit to the energy offset.

| Parameter | Value | 95% Confidence Bounds | |
|---|---|---|---|
| | | Lower | Upper |
| $p_1$ | -0.2056 | -0.3392 | -0.07191 |
| $p_2$ | 0.07694 | 0.05189 | 0.102 |
| $p_3$ | $-3.184 \cdot 10^{-4}$ | $-1.698 \cdot 10^{-3}$ | $1.061 \cdot 10^{-3}$ |
| $p_4$ | $-2.849 \cdot 10^{-5}$ | $-4.97 \cdot 10^{-5}$ | $-7.269 \cdot 10^{-6}$ |

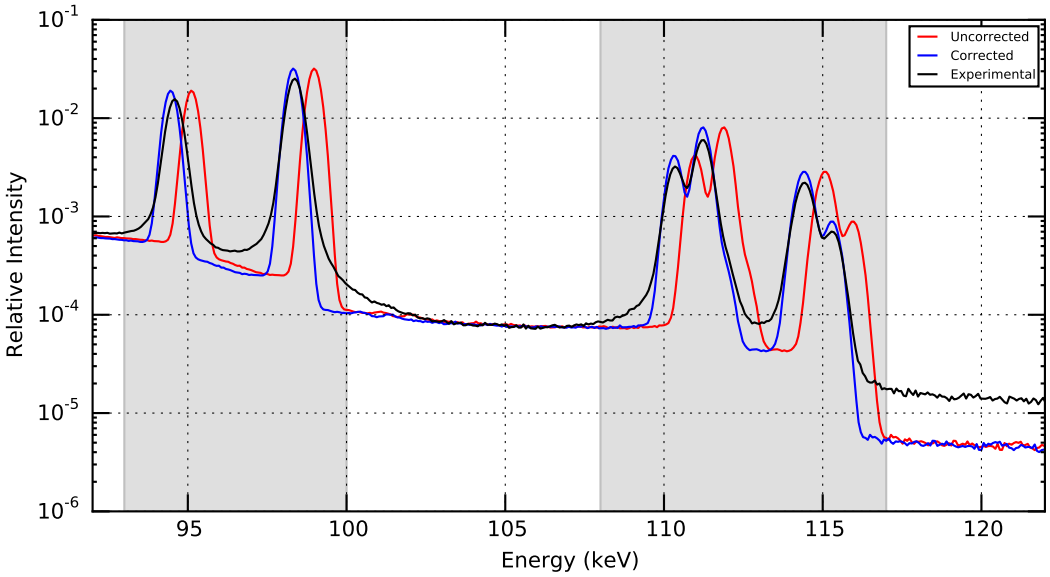


**Figure 5.6:** Uncorrected and corrected pulse height spectrum showing MCNP energy shift. Data presented in red represents the modeled uncorrected spectrum while the corrected spectrum is shown in green. Measured data is shown in blue. Higher deviations from the measured data are present at between 94 and 99 keV and 110 and 115 keV which correspond the $K_\alpha$ and $K_\beta$ peaks, respectively.

As shown in Figure 5.6, the MCNP generated pulse height spectrum over predicts the energy of the peaks by a small margin. The uncorrected spectrum exhibits a greater departure from the measured data approximately 95 and 99 keV and 115 and 118 keV. These energies correspond to the higher energy tails of the $K_\alpha$ and $K_\beta$ peaks. This shift was corrected by reassigning the Stage 1 energy bins to the correct values according to Equation 5.1.

# Chapter 6

# Model Validation

## 6.1 Validation Simulations

A series of simulations were completed using the MCNP model of the HKED system and compared to data obtained at Oak Ridge National Laboratory's Radiochemical Engineering Development Center. The purpose of these simulations was to predict the system response to samples of varied concentrations in both the XRF and KED detectors. Pulse height spectra were represented as relative intensity by normalizing the spectra to the continuum-subtracted intensity of the $^{109}$Cd gamma-ray peak at 88 keV. This was done by first adjusting the intensity of the $^{109}$Cd peak in the Stage 2 simulations to match those of the measured data. Then the modeled and measured continuum was determined for each spectrum's $^{109}$Cd peak and the area under the peak calculated. Each energy bin was then normalized to this value

Comparisons of the model to measured data were made by calculating the absolute fractional difference of the two data sets. This was done by directly comparing each measured and modeled energy bin intensity by the following relation:

$$Abs.\ Frac.\ Diff. = \left| 1 - \frac{Model}{Measured} \right| \tag{6.1}$$

### 6.1.1 K X-ray Fluorescence Cases

#### Uranium Cases

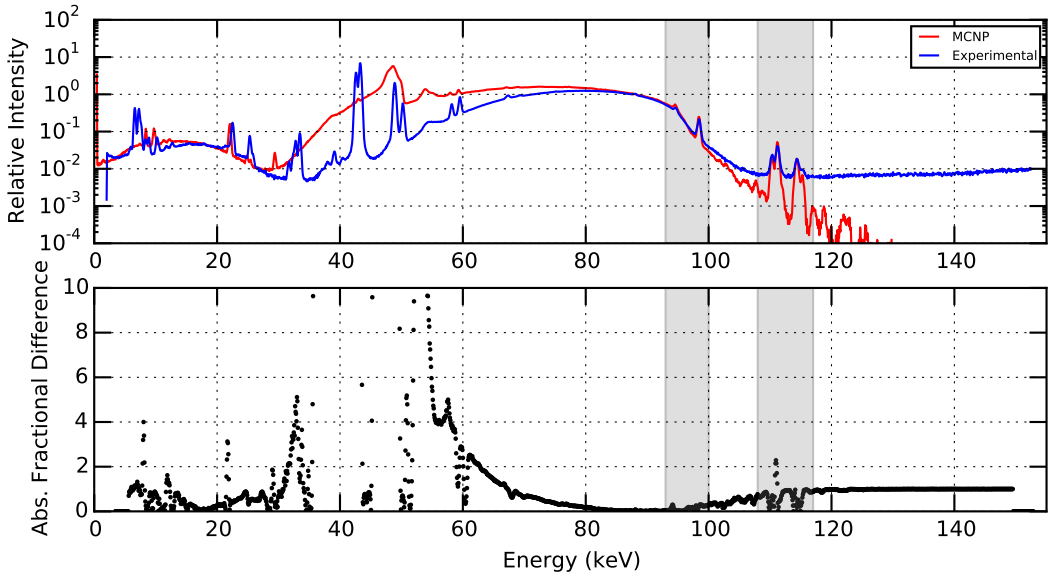


**Figure 6.1:** XRF pulse height spectrum of 1.072 $^{g}/_{L}$ uranium solution and residuals. Large differences between the model and the measured data are present at energies below 80 keV. The model does accurately model the $K_{\alpha 1}$ and $K_{\alpha 2}$ peak intensities but not the area between the $K_{\beta}$ peaks. The difference here is due to the lack of photons from the x-ray source.

Figure 6.1 shows the pulse height tally response to a 1.072 $^{g}/_{L}$ solution of uranium. The model does not accurately follow the measured data below 80 keV however, the model does accurately capture the K x-ray emission peaks between 90 and 100 keV ($K_{\alpha}$ x-rays) and 107 to 115 keV ($K_{\beta}$ x-rays). There is a discrepancy in the areas between the regions of interest and the $K_{\beta}$ peaks, but it is believed this is due the modeled x-ray source.

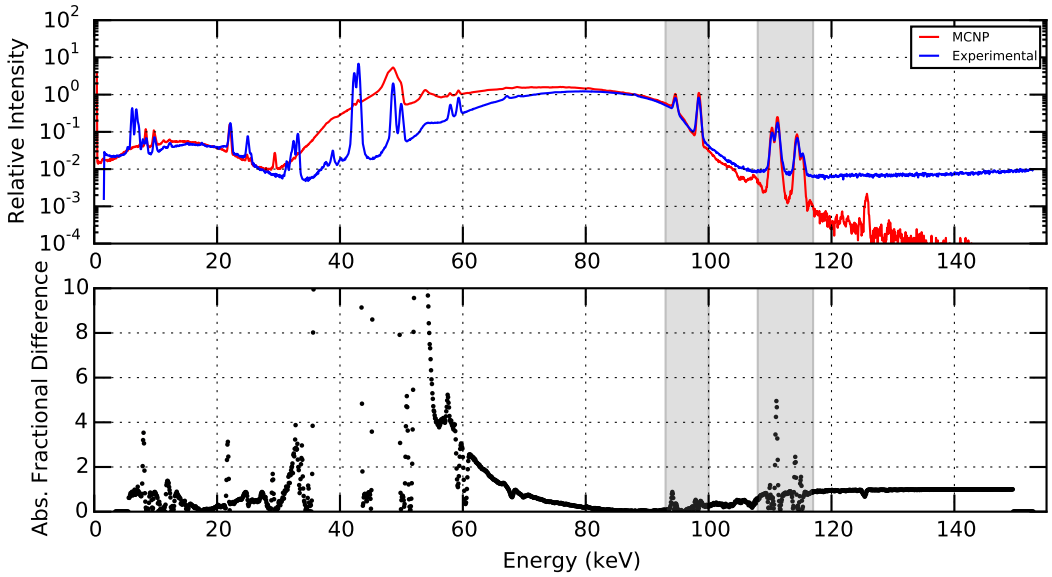


**Figure 6.2:** XRF pulse height spectrum of 5.459 $^{g}/_{L}$ uranium solution and residuals. As in the 1.072 $^{g}/_{L}$ case the model does not capture the gadolinium peaks between 40 and 50 keV. The $K_{\alpha 1}$ and $K_{\alpha 2}$ peaks are correctly modeled, however the energy ranges between the peaks are under represented but not as much as the 1.072 $^{g}/_{L}$ case.

Figure 6.2 shows the pulse height tally response to a 5.459 $^{g}/_{L}$ solution of uranium. Again, the model does not accurately follow the measured data below 80 keV however, the model does accurately capture the K x-ray emission peaks in the regions of interest.

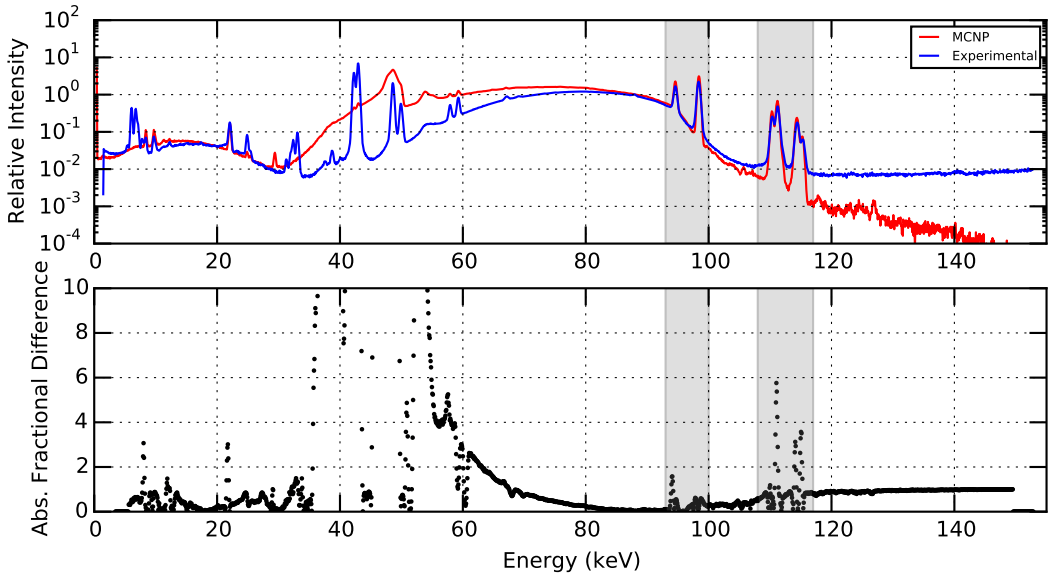


**Figure 6.3:** XRF pulse height spectrum of 15.895 $^{g}/_{L}$ uranium solution and residuals. Lower energies are still over represented in the model, however the energy ranges between the x-ray peaks are closer to the measured data. $K_{\alpha}$ and $K_{\beta}$ peak shapes are accurately represented.

Figure 6.3 shows the pulse height tally response to a 15.895 $^{g}\_L$ solution of uranium. K x-ray peak intensities and shapes are captured and the continuum between the ROIs and the $K_{\beta}$ peaks more closely matches the measured data.

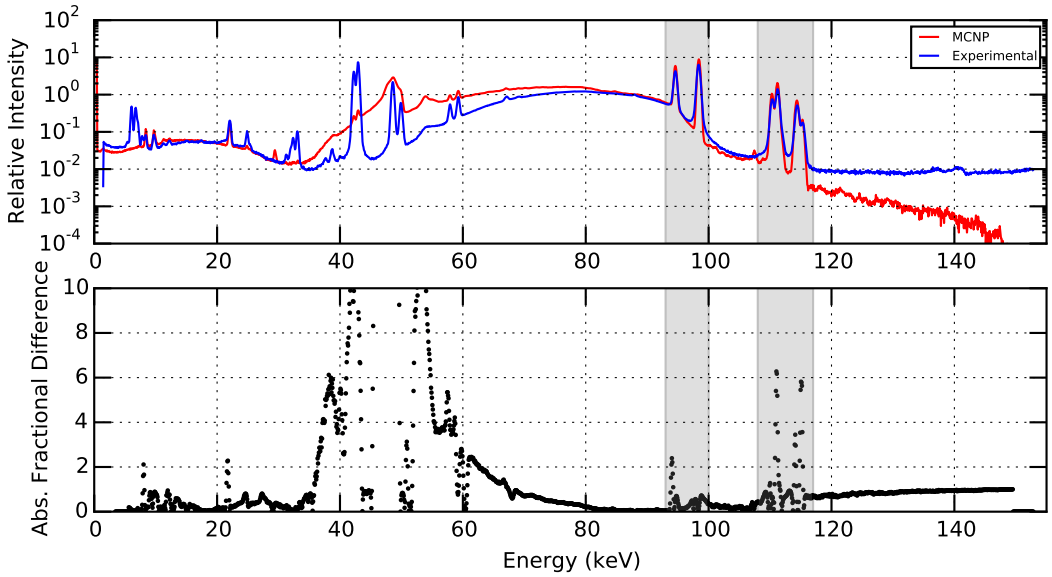


**Figure 6.4:** XRF pulse height spectrum of 48.12 $^{g}/_{L}$ uranium solution and residuals. Accuracy of gadolinium x-ray peaks are approaching the measured data. The model is capturing the intensity and accuracy of the K x-ray peaks.

Figure 6.4 shows the pulse height tally response to a 48.12 $^{g}/_{L}$ solution of uranium. K x-ray peak intensities and shapes are captured and the continuum between the ROIs and the $K_{\beta}$ peaks remains similar to that of the 15.895 $^{g}/_{L}$ case.

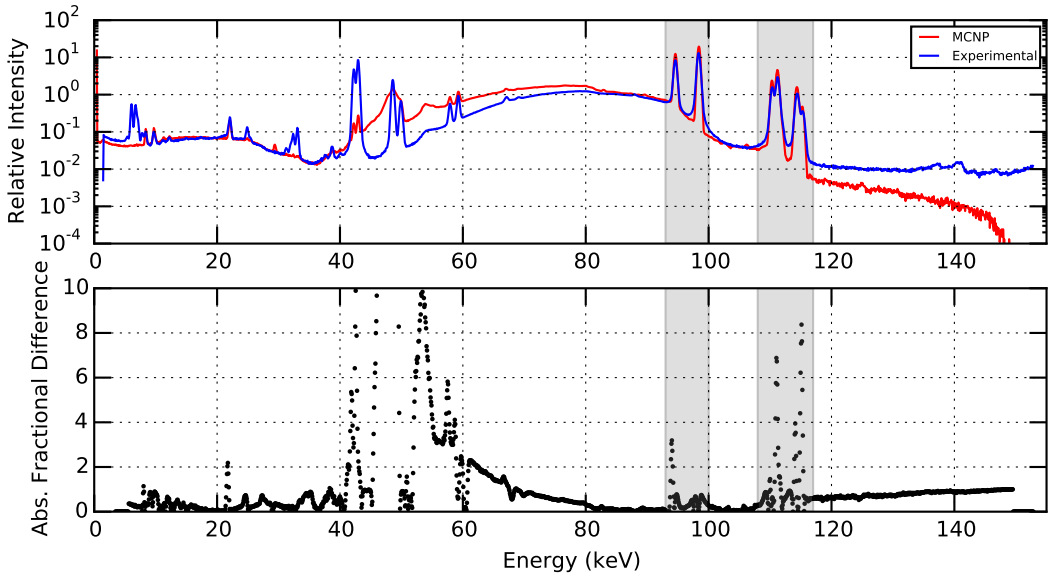


**Figure 6.5:** XRF pulse height spectrum of 107.1 $^{g}/_{L}$ uranium solution and residuals. K x-ray peaks in the regions of interest are accurately captured.

Figure 6.5 shows the pulse height tally response to a 107.1 $^{g}/_{L}$ solution of uranium. K x-ray peak intensities and shapes are captured and the continuum between the ROIs matches that of the measured data, however there is a discrepancy between the $K_{\beta}$ peaks. The gadolinium x-ray peaks between 40 and 50 keV deviate from the model. This may be due to an inconsistency in the thickness of the gadolinium filter.

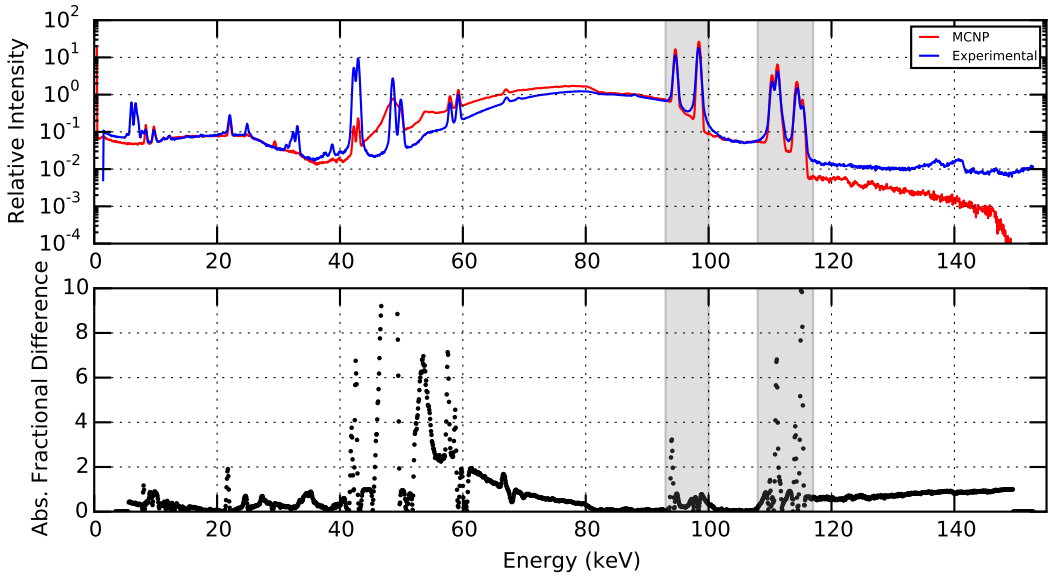


**Figure 6.6:** XRF pulse height spectrum of 160.8 $^{g}/_{L}$ uranium solution and residuals. Differences from the measured data remain below 80 keV but the K x-ray emissions are accurately modeled.

Figure 6.6 shows the pulse height tally response to a 160.8 $^{g}/_{L}$ solution of uranium. The lower end of the spectrum, specifically the gadolinium x-ray peaks between 40 and 50 keV, is more closely matching the measured data.

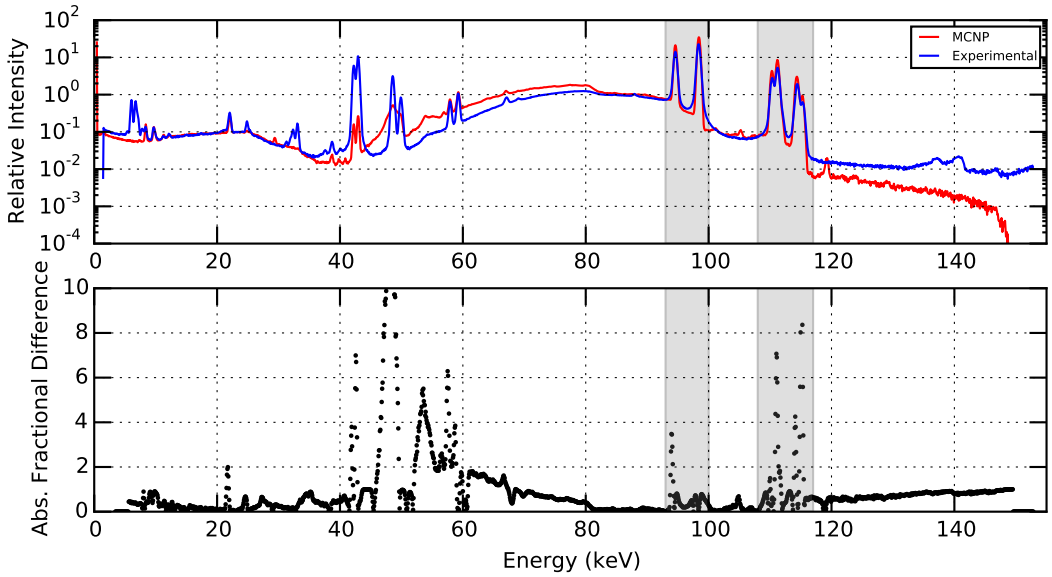


**Figure 6.7:** XRF pulse height spectrum of 214.5 $^{g}/_{L}$ uranium solution and residuals. As in the 160.8 $^{g}/_{L}$ case, differences from the measured data remain below 80 keV but the K x-ray emissions are accurately modeled.

Figure 6.7 shows the pulse height tally response to a 214.5 $^{g}/_{L}$ solution of uranium. Again, the lower end of the spectrum below 80 keV, specifically the gadolinium x-ray peaks between 40 and 50 keV, is more closely matching the measured data.

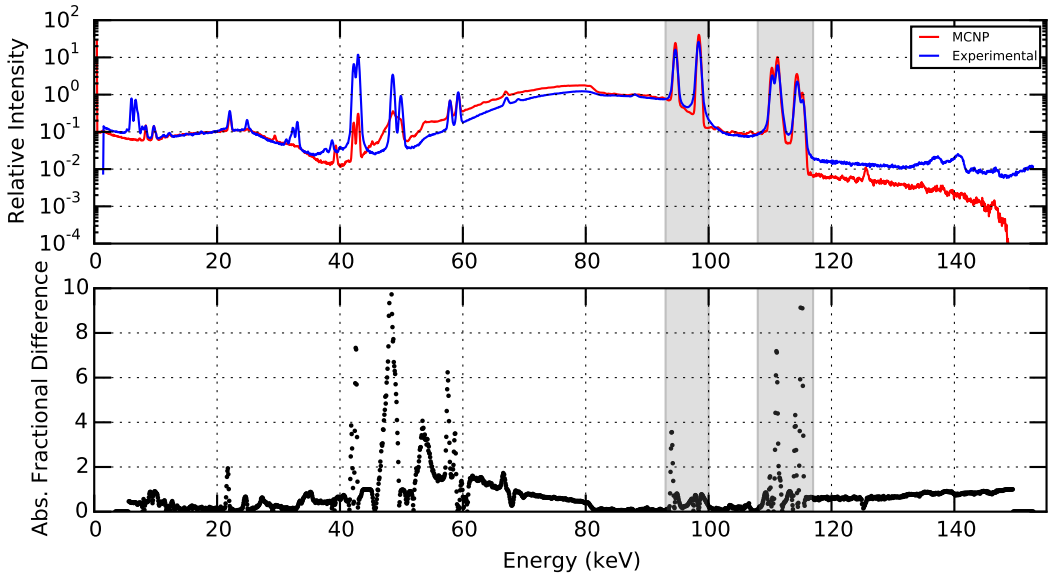


**Figure 6.8:** XRF pulse height spectrum of 268.4 $^{g}/_{L}$ uranium solution and residuals. The model is approaching the measured data below 80 keV but is more accurate in the regions of interest.

Figure 6.8 shows the pulse height tally response to a 268.4 $^{g}/_{L}$ solution of uranium. Again, the lower end of the spectrum below 80 keV, specifically the gadolinium x-ray peaks between 40 and 50 keV, more closely matches the measured data.

As shown in the above figures there are inconsistencies in the model below 80 keV, especially at the gadolinium x-ray emission peaks between 40 and 50 keV. The modeled K x-ray emissions increase as the uranium concentration increases and accurately models the shape of the peaks, however the intensity of the peaks appear to be over exaggerated. The inconsistencies in the lower end of the spectrum could be due to the x-ray source not being true to the spectrum found in the HKED instrument's x-ray source below 80 keV and a misconfiguration of the gadolinium foil. Unfortunately the foil's fragility prevented its removal from the XRF collimator for measurement.

### Uranium and Plutonium Cases

In addition to the uranium only aqueous cases, three other simulations were completed with plutonium containing samples. Four separate simulations were completed,

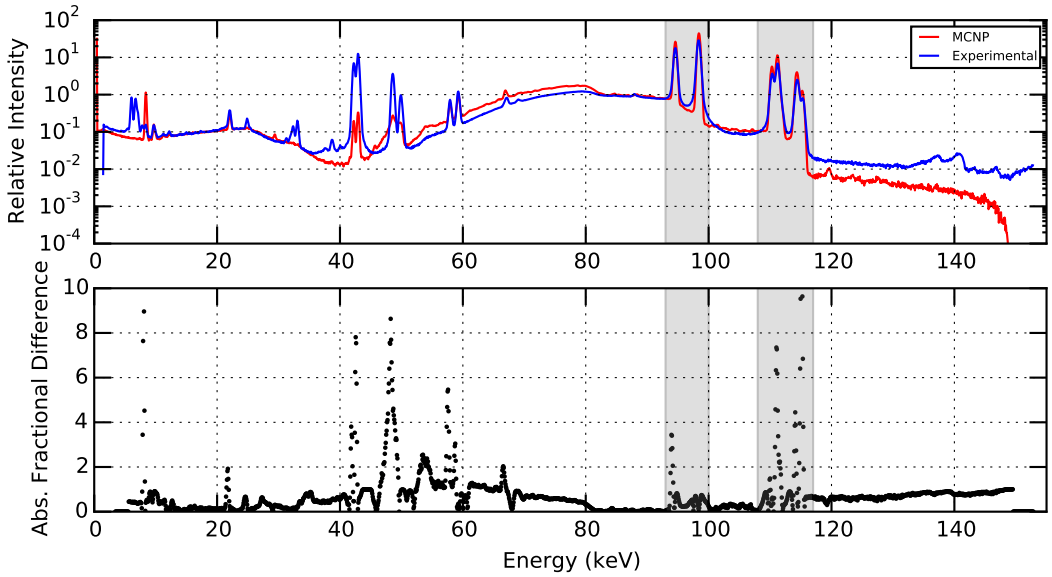


**Figure 6.9:** XRF pulse height spectrum of 323.7 $^{g}/_{L}$ uranium solution and residuals. The model is capturing the K x-ray emissions and is approaching the measured data below 80 keV.

however only three measurements were made for comparison. The mass ratios of uranium to plutonium varied between 103:1 and 82.81:1.

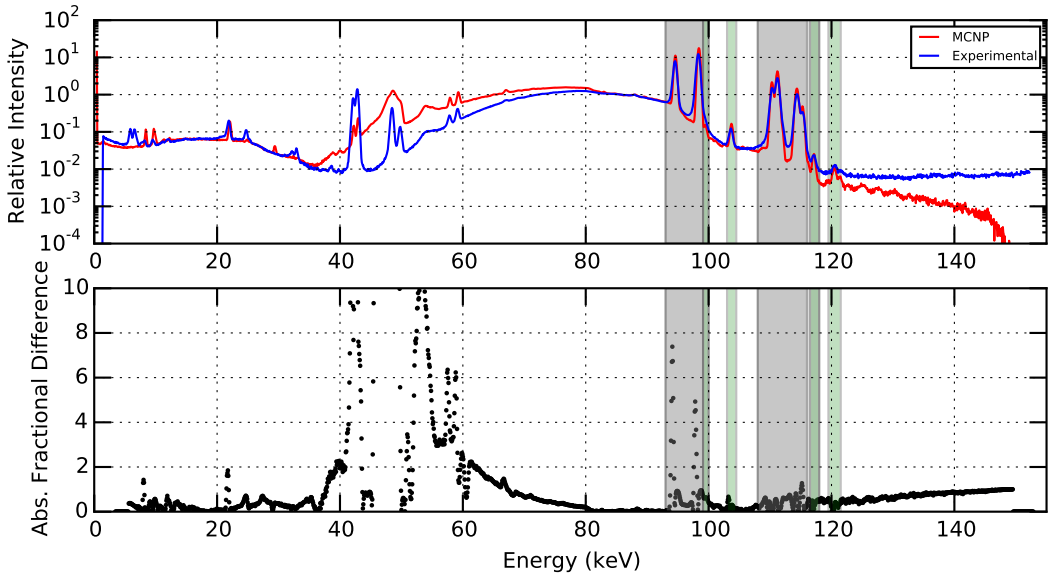


**Figure 6.10:** XRF pulse height spectrum of 107.5 $^{g}/_{L}$ uranium solution and plutonium at 103:1 mass ratio of uranium to plutonium. As in the uranium cases the model begins to deviate from the measured data below 80 keV but dues accurately capture the uranium K x-ray emissions. The plutonium $K_{\alpha 1}$ peak is captured however the $K_{\alpha 2}$ peak is lost in the higher energy shoulder of the uranium $K_{\alpha 1}$. The plutonium $K_{\beta}$ peaks are present and relatively well represented.

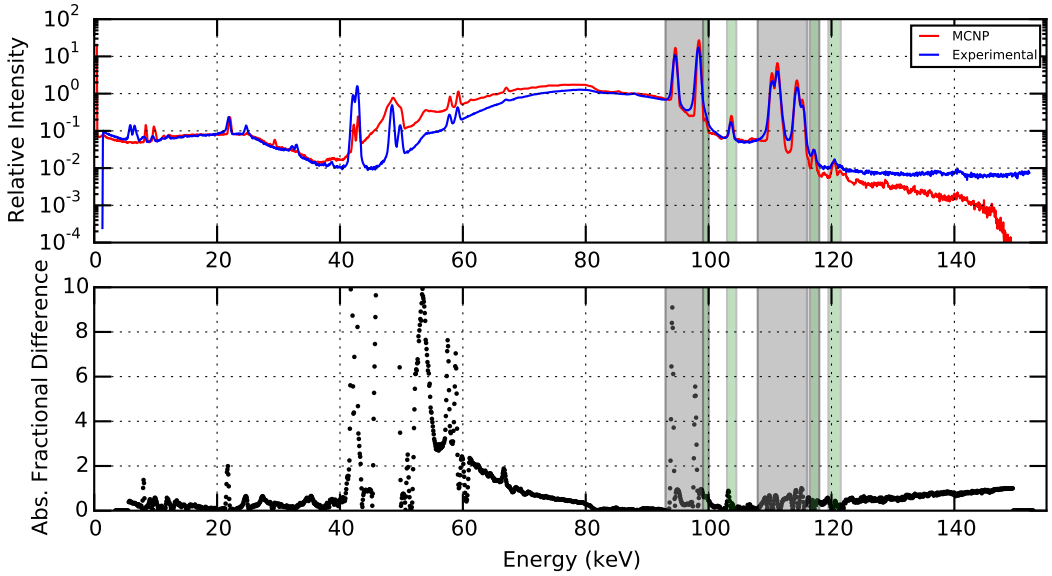


**Figure 6.11:** XRF pulse height spectrum of 160.8 $^{g}/_{L}$ uranium solution and plutonium at 103:1 mass ratio of uranium to plutonium. Deviation of the model from measured data remains relatively constant in energy regions of interest. Lower energies are more accurately represented between 42 and 59 keV.

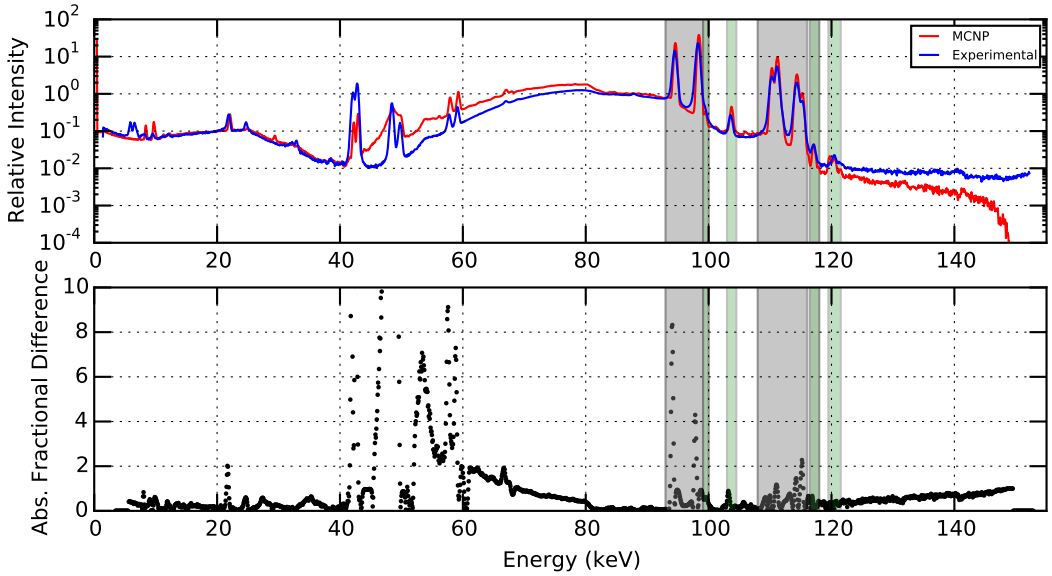


**Figure 6.12:** XRF pulse height spectrum of 250 $^{g}/_{L}$ uranium and plutonium solution and residuals. Deviation of the model from measured data remains relatively constant in energy regions of interest. Lower energies are more accurately represented between 42 and 59 keV.

Like the uranium only cases, the uranium-plutonium cases suffer from a deviation of the model below 80 keV. The K x-ray emissions of both the uranium and plutonium $K_{\alpha 1}$ are visible. The plutonium $K_{\beta}$ peaks are faint in comparison to the uranium $K_{\beta}$ peaks, however they are clearly visible in the spectrum. The same inconsistencies below 80 keV are present but the model is relatively accurate in the region of interest.

### 6.1.2 K X-ray Peak Intensity Analysis

In addition to the channel by channel comparison which checked for the correct shape of the whole spectrum an analysis was completed on the peak intensity for both $K_{\alpha}$ and $K_{\beta}$ peaks. This was to determine if the total intensities for each peak was correctly modeled. The areas under each peak for both $K_{\alpha}$ peaks were calculated individually but the proximity to of the $K_{\beta}$ peaks to one another, specifically the $K_{\beta 13}$ and $K_{\beta 24}$ peaks, required calculating their intensities as doublets or groups of two peaks.

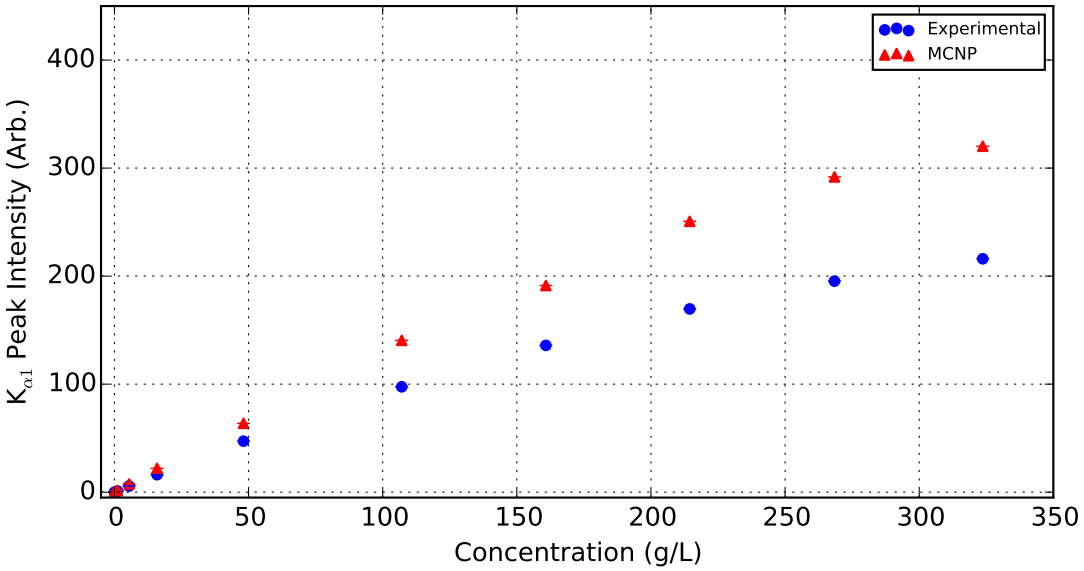


**Figure 6.13:** $K_{\alpha 1}$ peak intensities as a function of uranium concentration. Below 50 keV the model matches the measured peak intensity relatively well, however a significant deviation occurs at higher energies.

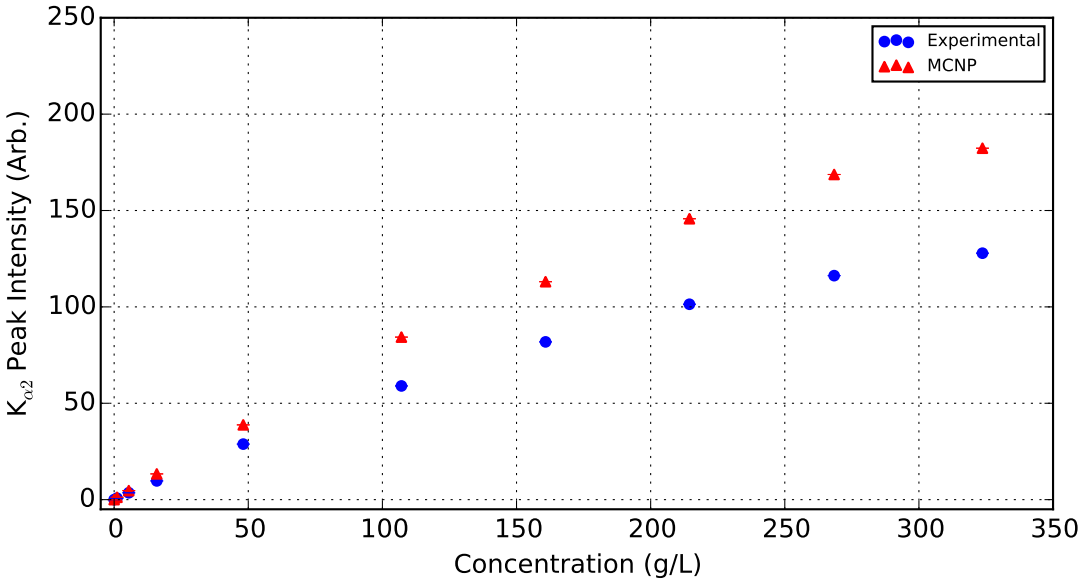


**Figure 6.14:** $K_{\alpha 2}$ peak intensities as a function of uranium concentration. Again, a significant deviation occurs at higher energies, while lower energies more closely match the measured data.

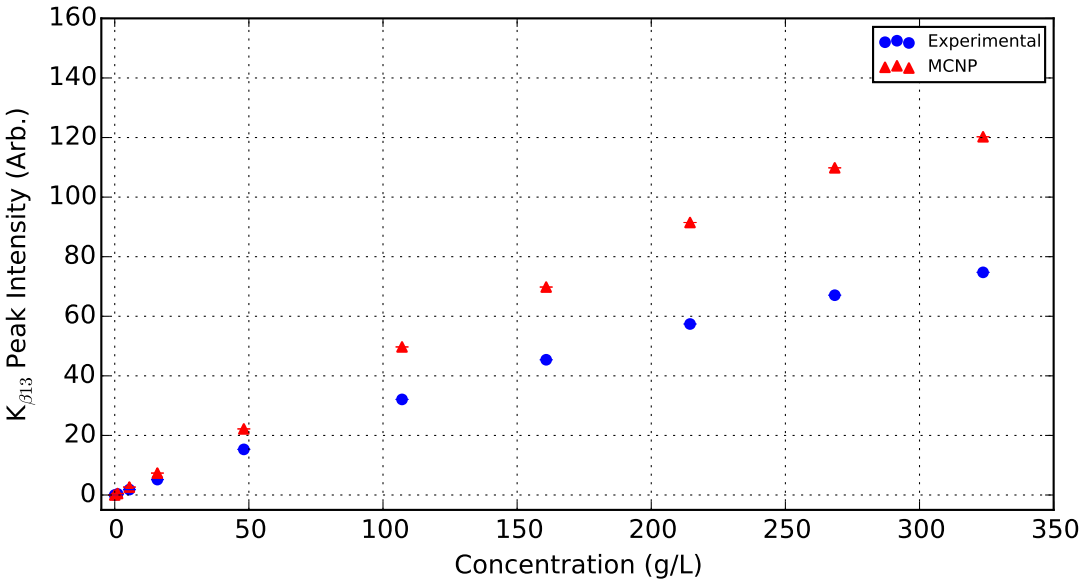


**Figure 6.15:** $K_{\beta 13}$ peak intensities as a function of uranium concentration. The same deviation is seen above 50 keV.

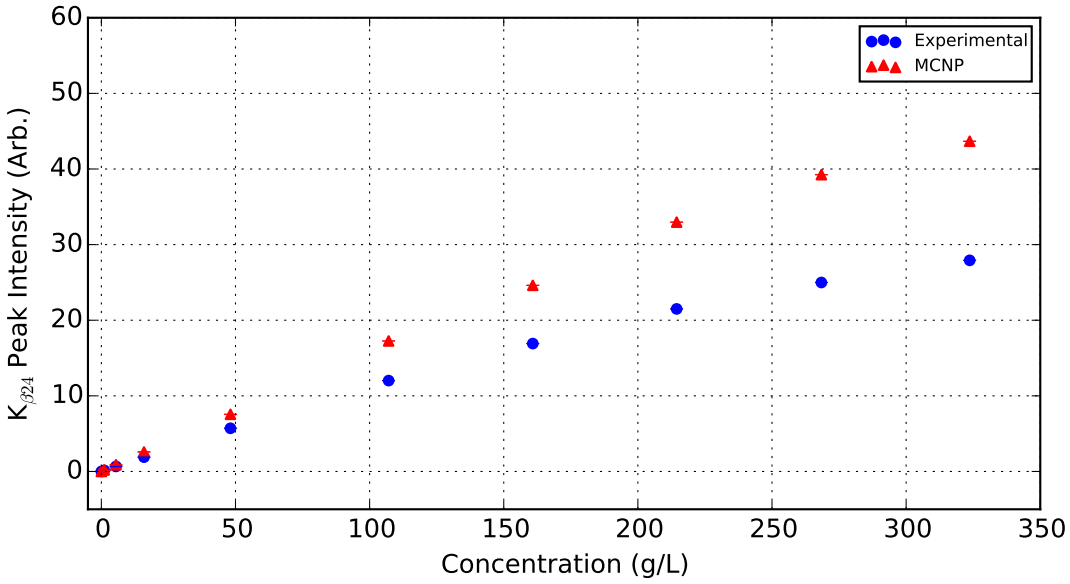


**Figure 6.16:** $K_{\beta 24}$ peak intensities as a function of uranium concentration. As in the other K x-ray emissions, a significant deviation occurs above 50 keV.

The peak intensities for the $K_\alpha$ and $K_\beta$ peaks match the modeled data well for the 1.072, 5.459, and 15.895 $^g/_L$ cases. A noticeable deviation begins to manifest as the concentration increases past 50 $^g/_L$ with a maximum deviation of approximately 38% at the highest concentration case. This is most likely due to inconsistencies in the MCNP photon libraries along with the energy shift issue. This inconsistency is explored further in Section 6.2.

An identical analysis was performed on the uranium and plutonium mixed solutions to determine the plutonium peak intensities. Four simulations were completed, however measured data only existed for three cases. These are presented below.

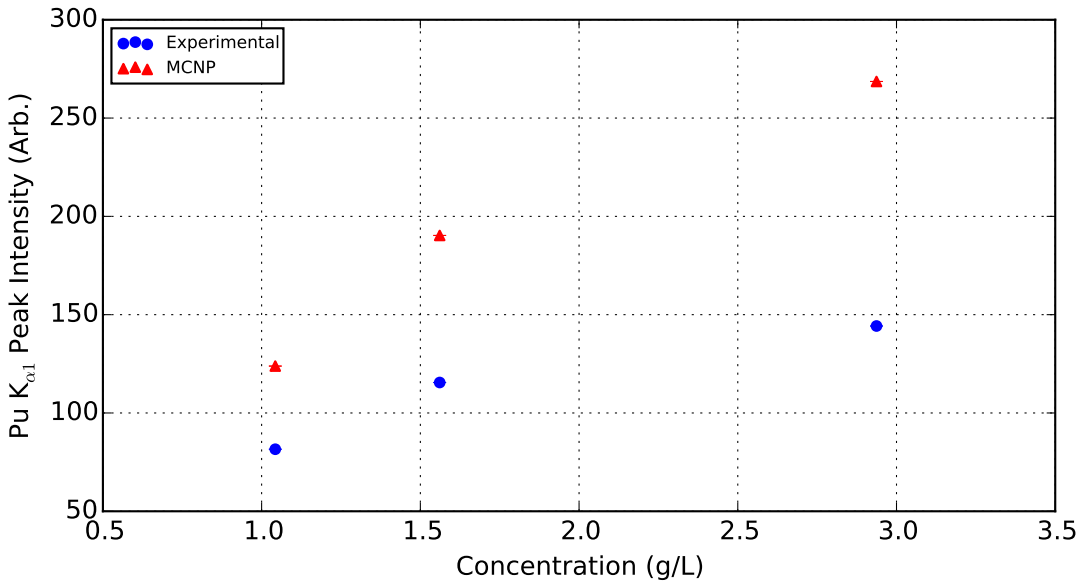


**Figure 6.17:** $K_{\alpha 1}$ peak intensities as a function of plutonium concentration. MCNP modeled response increases in magnitude as concentration of plutonium increases.

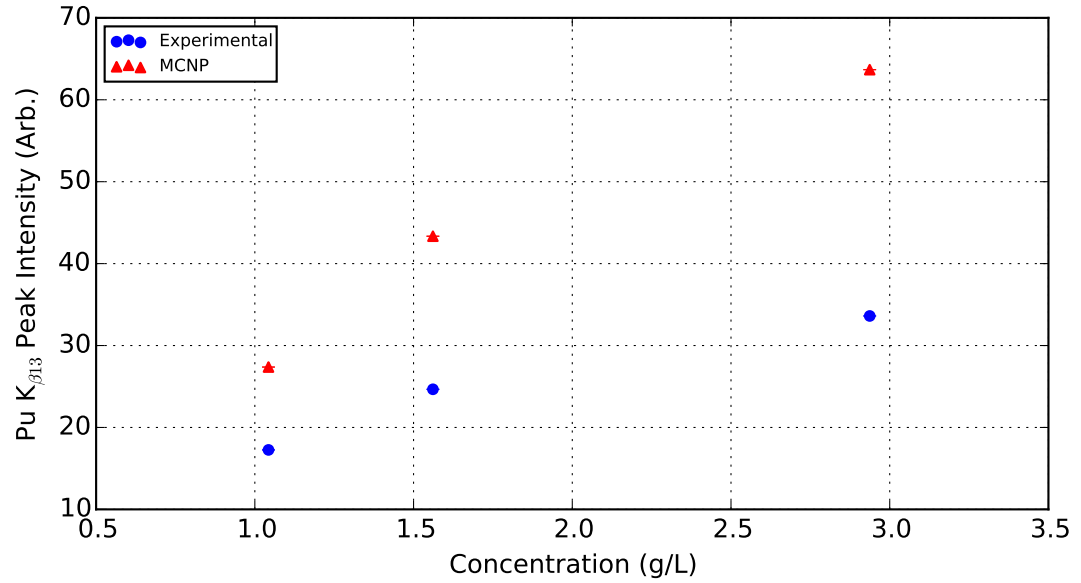


**Figure 6.18:** $K_{\beta 13}$ peak intensities as a function of plutonium concentration. As in the other aqueous solutions, the MCNP modeled peak intensity deviation from the measured data increases with concentration.

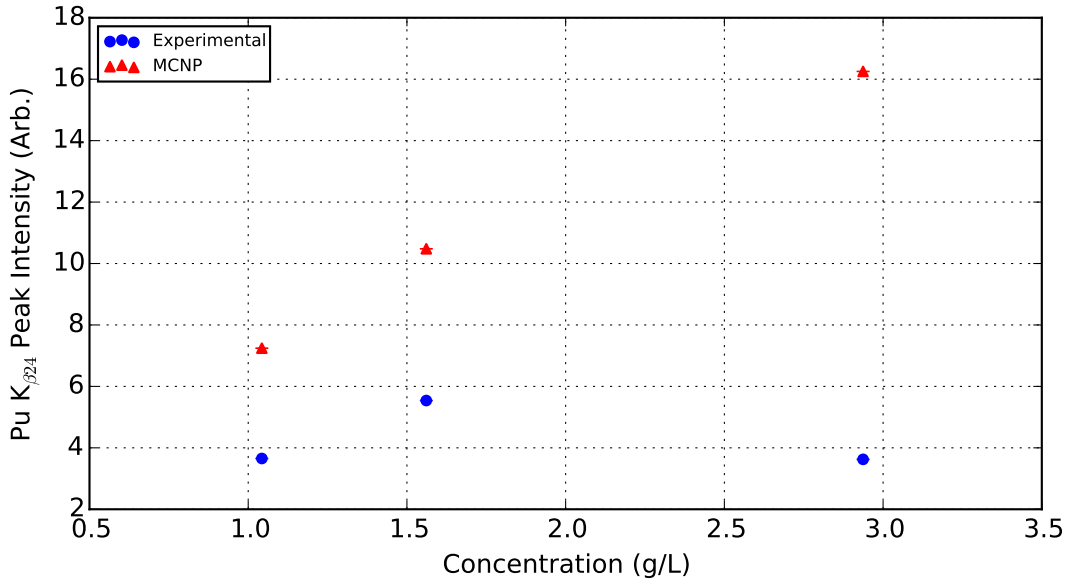


**Figure 6.19:** $K_{\beta 24}$ peak intensities as a function of plutonium concentration. Lower concentration peak intensities show a consistent offset, while the higher concentration case shows a substantial increase in the deviation.

Figures 6.17 to 6.19 depict the continuum subtracted peak intensity for plutonium K x-ray peaks. The $K_{\alpha 2}$ peak was not quantifiable in either the modeled and measured spectra.

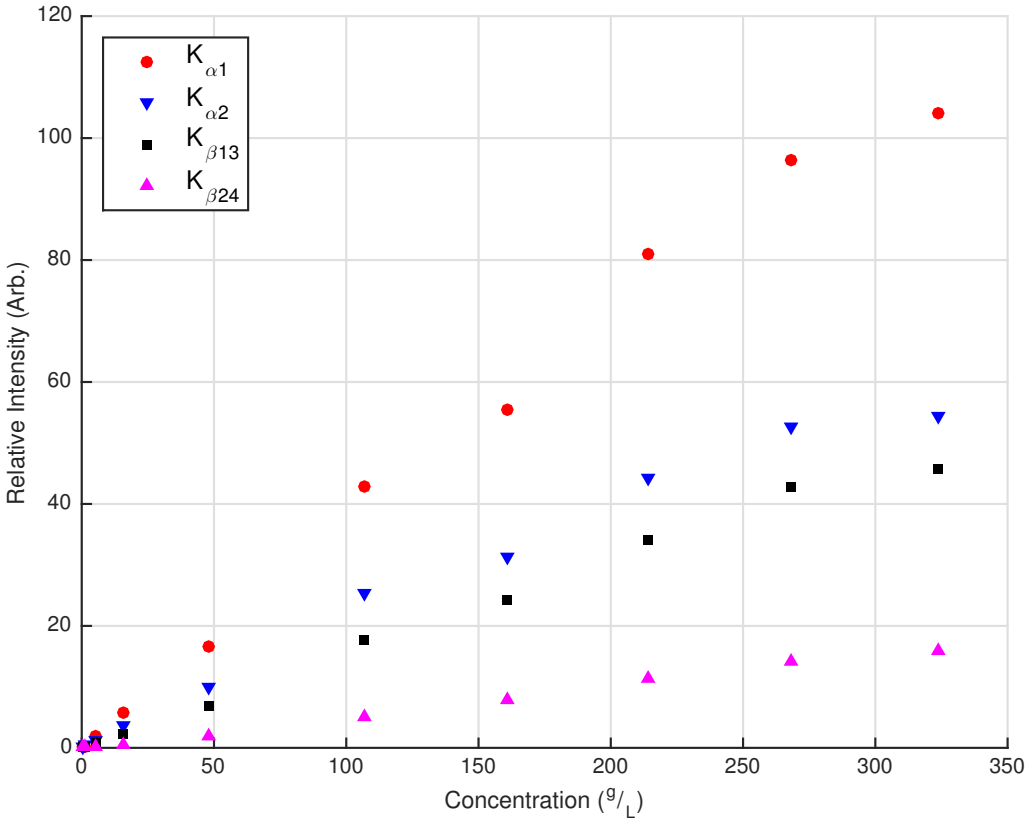


**Figure 6.20:** Differences between the measured and modeled K x-ray peak intensities as a function of uranium concentration in solution.

As shown in Figures 6.20 and 6.21, the difference is constantly increasing as the concentration of uranium increases. The largest difference is found in the $K_\alpha$ emission since it is the most prominent emission of uranium. These deviations are mostly linear across the range of concentrations. A further analysis was performed to calculate a fit to this data.

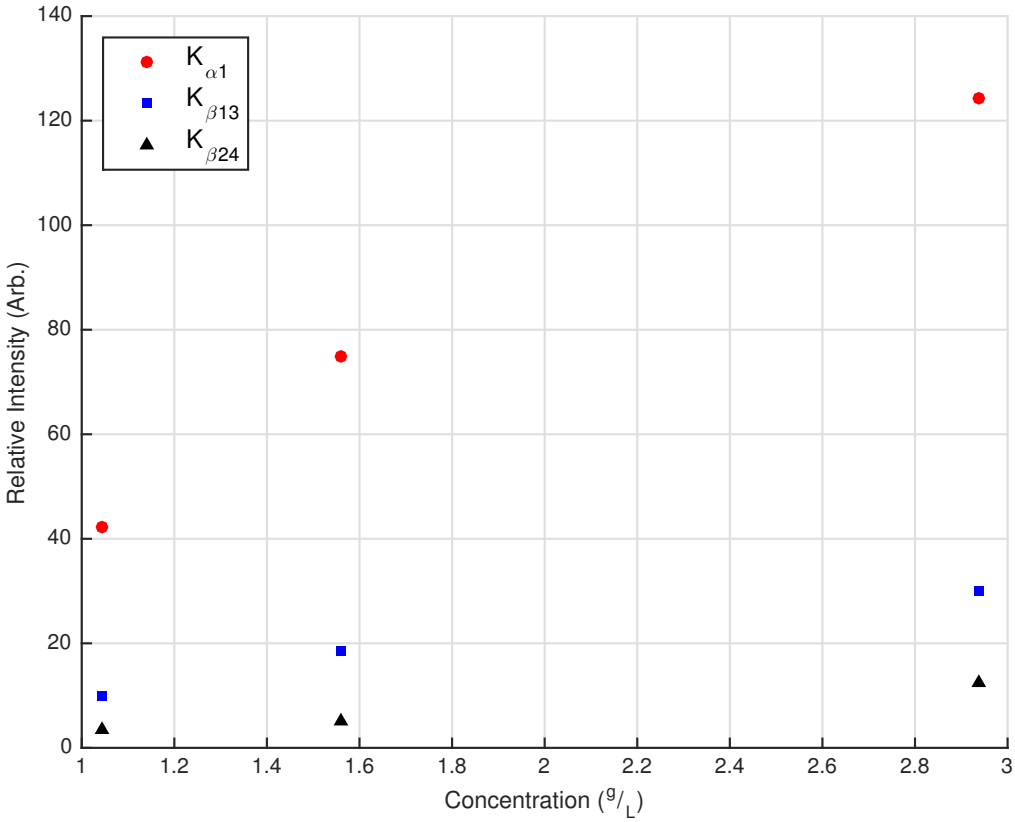


**Figure 6.21:** Differences between the measured and modeled K x-ray peak intensities as a function of plutonium concentration in solution.

A linear fit was applied to the K x-ray emission difference data to determine the relative intensity difference as a function of concentration. This was performed for each K x-ray emission. The fit follows the form:

$$Int.\,Diff. = a_1 \times Conc. + a_2 \tag{6.2}$$

The parameters for each K x-ray emission were calculated in Matlab and are given in Table 6.1.

**Table 6.1:** Linear fit parameters for the K x-ray intensity offset.

| Element | K x-ray | $a_1$ | $a_2$ | $R^2$ |
|---|---|---|---|---|
| U | $K_{\alpha 1}$ | 0.3424 | 1.282 | 0.9905 |
| U | $K_{\alpha 2}$ | 0.1821 | 1.459 | 0.982 |
| U | $K_{\beta 13}$ | 0.1499 | 0.284 | 0.9929 |
| U | $K_{\beta 24}$ | 0.05116 | -0.133 | 0.9944 |
| Pu | $K_{\alpha 1}$ | 0.6041 | -22.53 | 0.999 |
| Pu | $K_{\beta 13}$ | 0.1461 | -5.297 | 0.9982 |
| Pu | $K_{\beta 24}$ | 0.06844 | -4.686 | 0.9333 |

The rate at which the deviation increases is different for each K x-ray emission, however the deviation does follow a constant increase as the concentration increases.

### 6.1.3 K-edge Transmission Cases

Validation of the KED components, like the XRF simulations, consisted of comparison of MCNP-generated pulse height spectra against measured data of select concentrations of uranium. The intensity in each modeled channel was compared to the measured data as in the XRF simulations.

### Uranium Cases

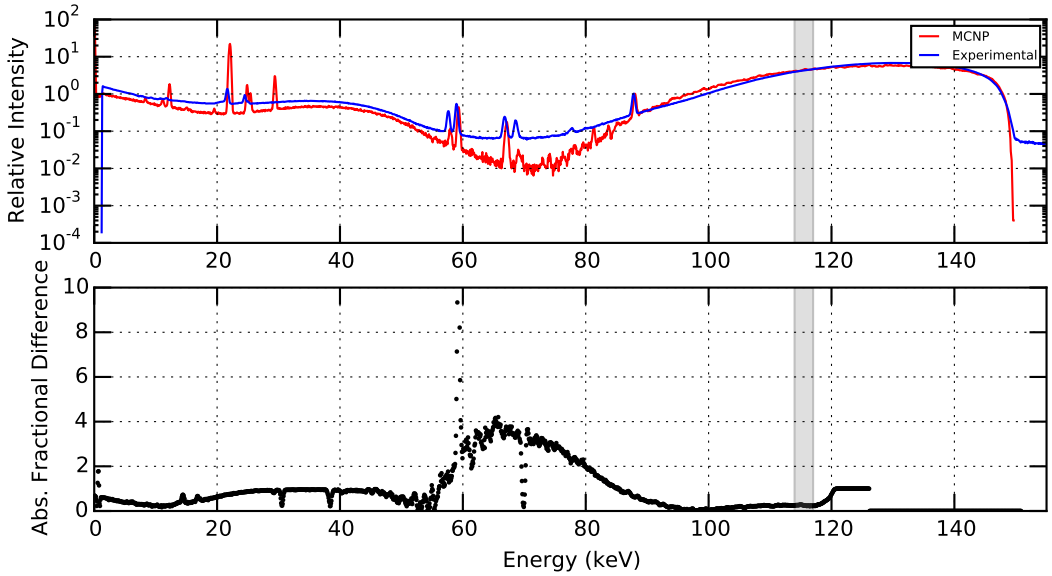


**Figure 6.22:** Pulse height spectrum of a K-edge transmission measurement of a 1.072 $^{g}/_{L}$ uranium sample. The uranium K-edge is not visible at this concentration.

The 1.072 $^{g}/_{L}$ case does not show a prominent K-edge drop. The $^{109}$Cd is visible and matches the measured data well. There is a deviation below the 88 keV peak especially around the tungsten K x-rays between 55 and 70 keV. This deviation in the measured data is attributed to an inaccurate model of the HKED x-ray source.

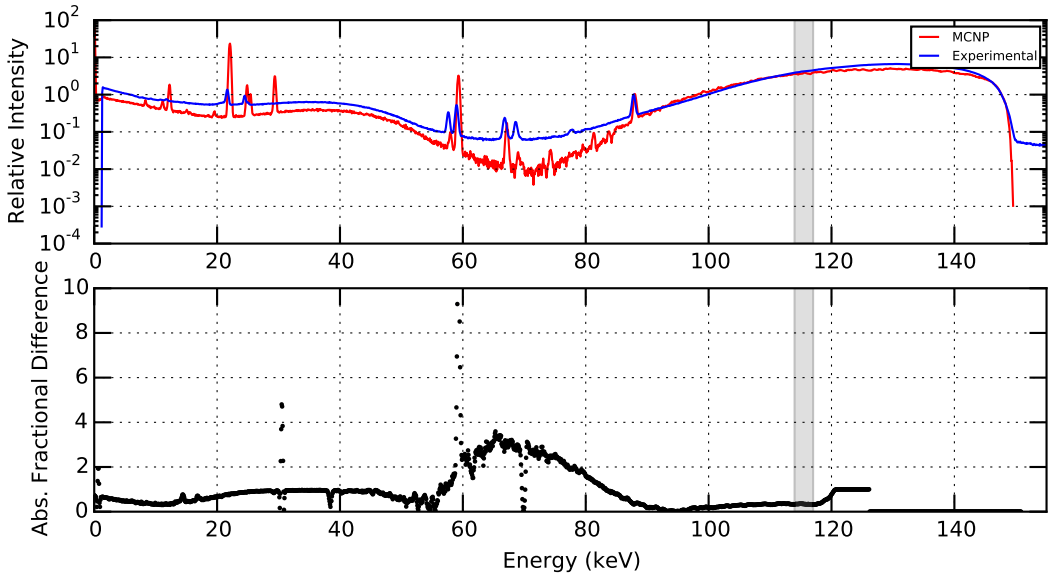


**Figure 6.23:** Pulse height spectrum of a K-edge transmission measurement of a 5.459 $^{g}/_{L}$ uranium sample. The uranium K-edge not visible at 115.6 keV.

As in the previous case, the 5.459 $^{g}/_{L}$ case does not show a prominent K-edge drop. The $^{109}$Cd is visible and matches the measured data well. The same spectrum drop is present at the tungsten K x-rays, however the $^{109}$Cd peak matches the measured data well.

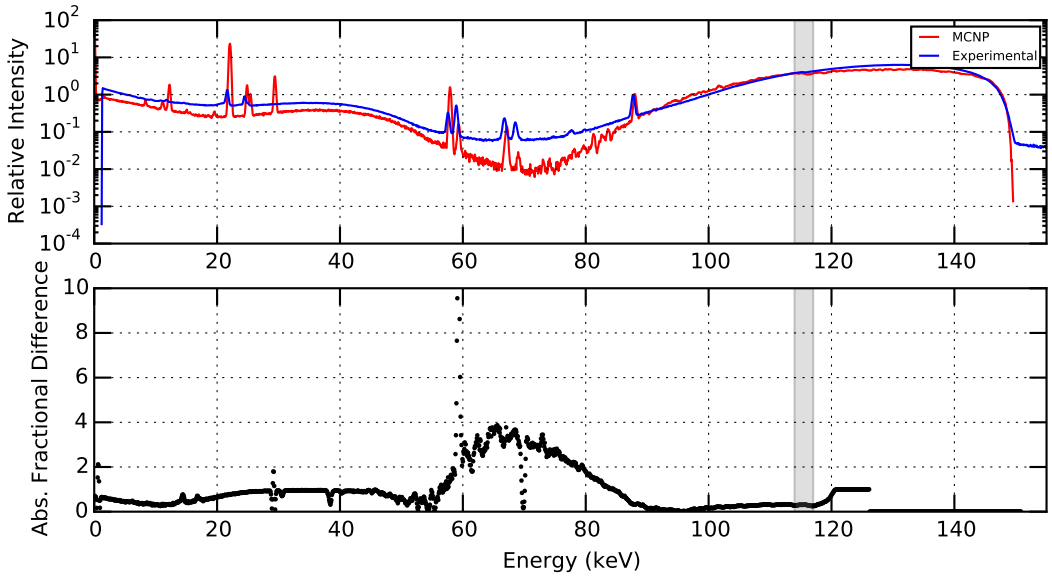


**Figure 6.24:** Pulse height spectrum of a K-edge transmission measurement of a 15.895 $^{g}/_{L}$ uranium sample. The uranium K-edge is slightly visible at 115.6 keV.

The 15.895 $^{g}/_{L}$ does show a slight K-edge drop at 115 keV, however it is very faint. The same spectrum drop is present at the tungsten K x-rays and the tungsten $K_{\alpha 2}$ peak is over estimated.

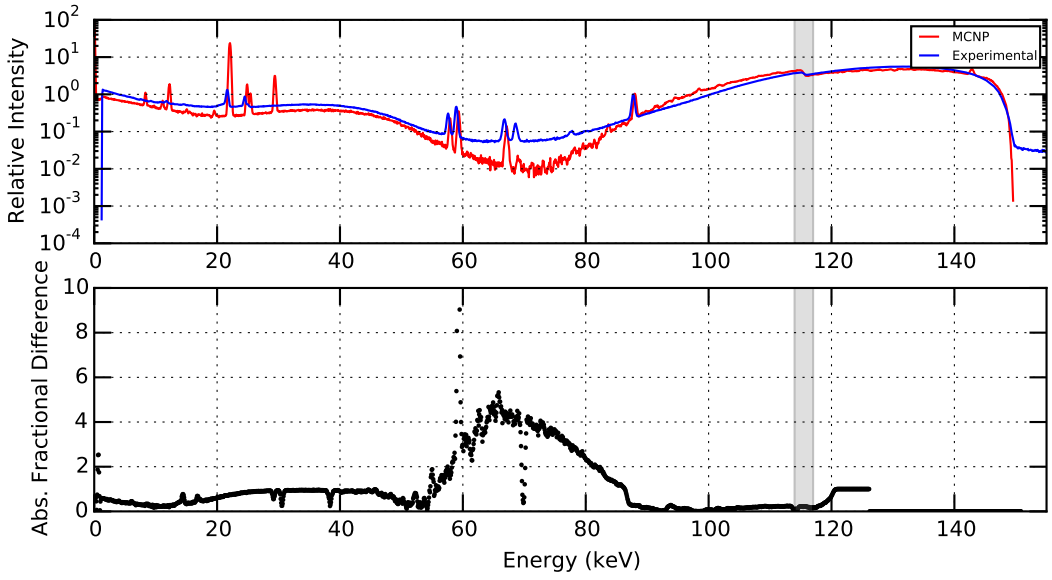


**Figure 6.25:** Pulse height spectrum of a K-edge transmission measurement of a 48.12 $^{g}/_{L}$ uranium sample. The uranium K-edge is visible at 115.6 keV.

Figure 6.25 contains a visible uranium K-edge at 115.6 keV. The same deviations at the tungsten x-rays are present.

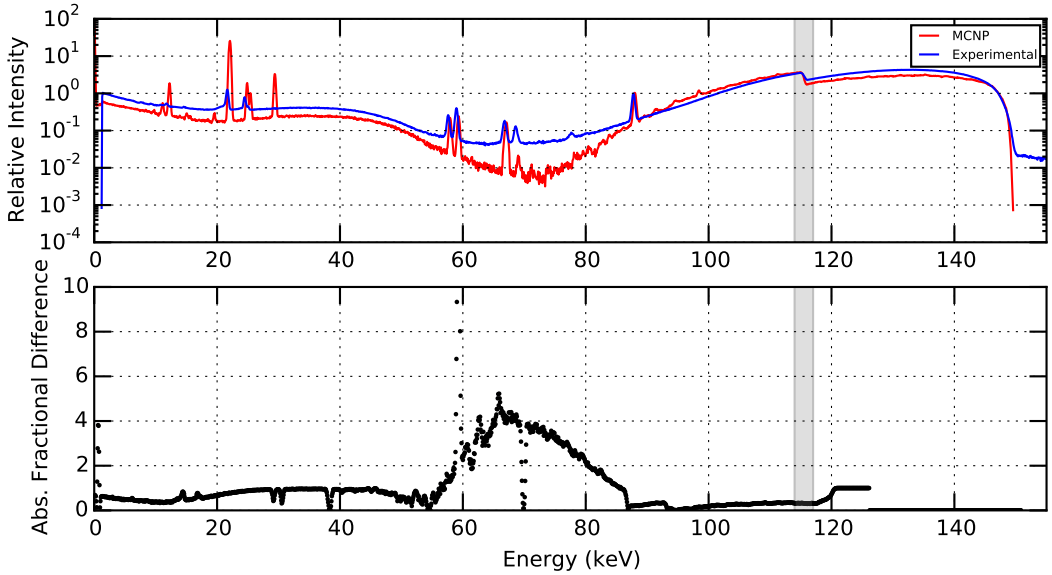


**Figure 6.26:** Pulse height spectrum of a K-edge transmission measurement of 1K-edge07.1 $^{g}/_{L}$ uranium. Model deviation from measured data is slightly greater at the uranium K-edge of 115.6 keV.

Figure 6.26 shows a deviation beginning at the uranium K-edge. The drop is slightly greater than the measured data. This would indicate that more photons at the uranium K-edge are being absorbed.

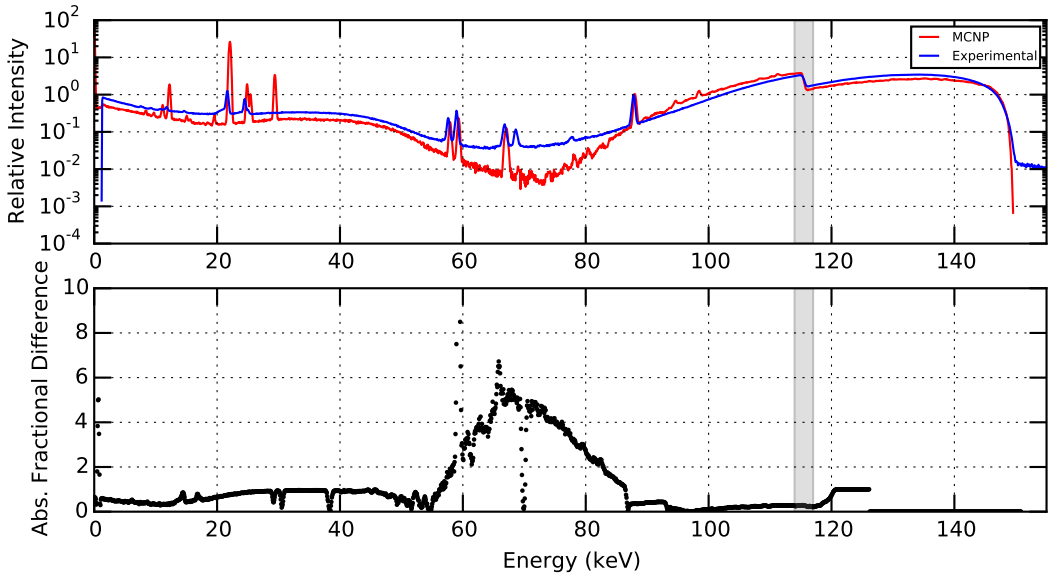


**Figure 6.27:** Pulse height spectrum of a K-edge transmission measurement of 160.8 $^{g}/_{L}$ uranium. Model deviation from measured data is slightly greater at the uranium K-edge of 115.6 keV.

The same over-prediction of the uranium K-edge drop is evident in Figure 6.27. The underestimation of the spectrum around the tungsten K x-rays between 55 and 70 keV is also evident.

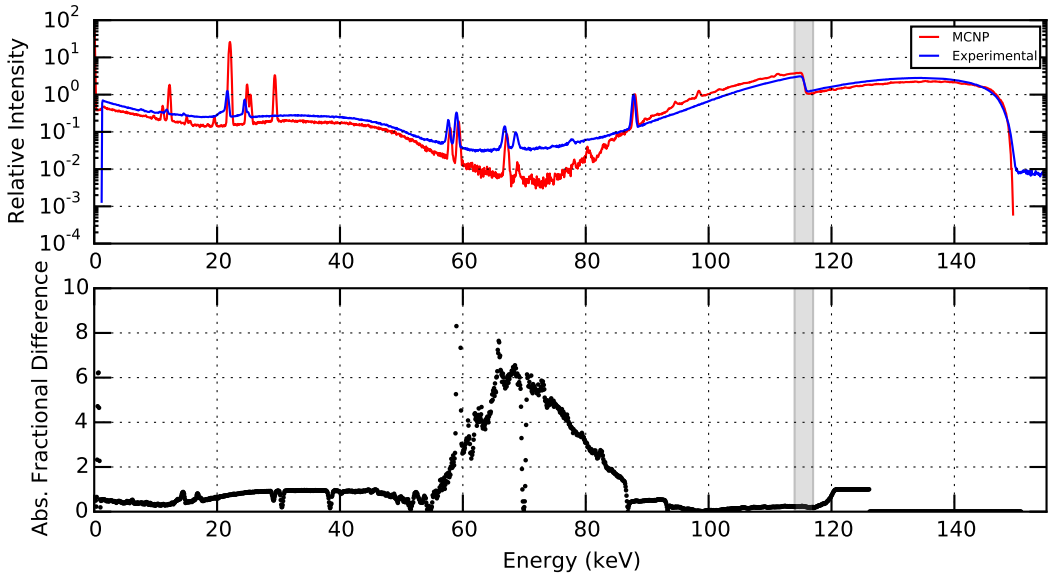


**Figure 6.28:** Pulse height spectrum of a K-edge transmission measurement of 214.5 $^{g}/_{L}$ uranium. Model deviation from measured data is slightly greater at the uranium K-edge of 115.6 keV.

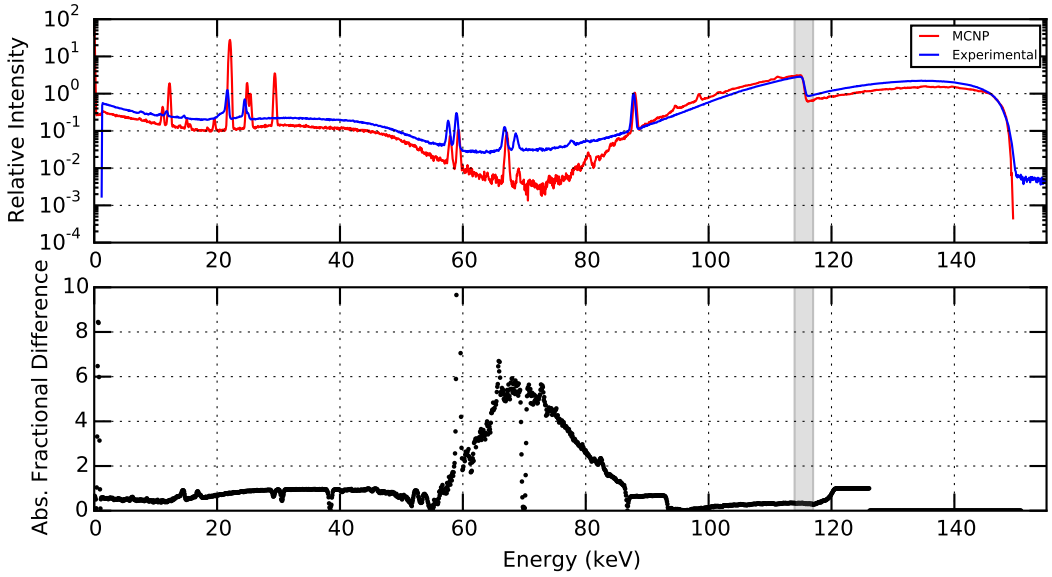


**Figure 6.29:** Pulse height spectrum of a K-edge transmission measurement of 268.4 $^{g}/_{L}$ uranium. Model deviation from measured data is slightly greater at the uranium K-edge of 115.6 keV.

Both the 214.5 and 268.4 $^{g}/_{L}$ cases show the over estimation of the K-edge drop. A prominent drop in the spectrum is still present around the tungsten K x-rays between 55 and 70 keV.

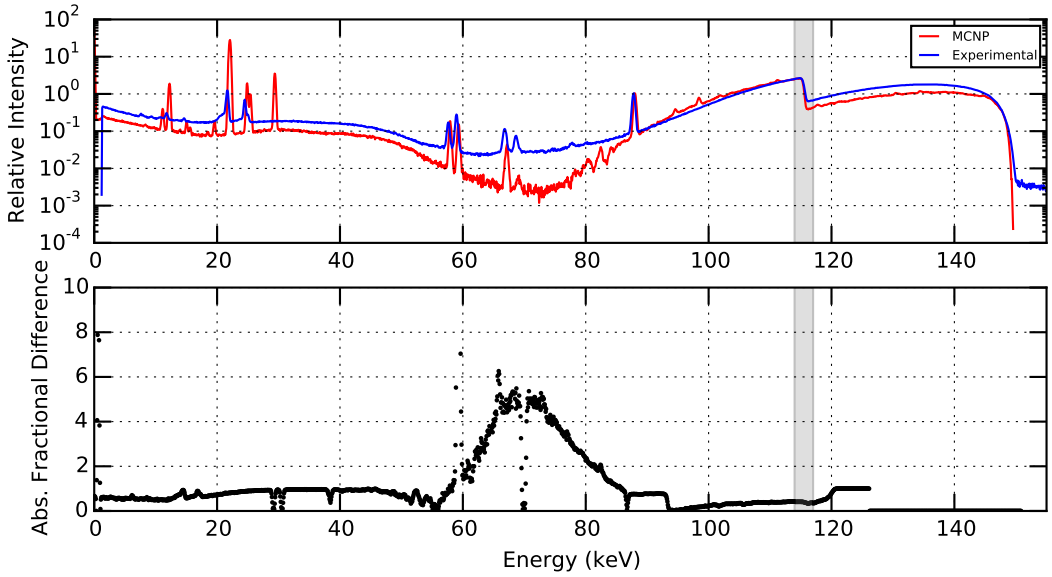


**Figure 6.30:** Pulse height spectrum of a K-edge transmission measurement of 323.7 $^{g}/_{L}$ uranium. Model deviation from measured data is slightly greater at the uranium K-edge of 115.6 keV.

Uranium's K-edge is prominent in the higher concentration case at 115.6 keV as shown in Figure 6.30. In each of the spectra the MCNP under-estimation is present around the tungsten x-rays and below the 88 keV $^{109}$Cd peak. The $^{109}$Cd x-rays between 10 keV and 30 keV did not match the measured data even though the x-ray intensities were configured to match the data from the NNDC decay database (38).

### Uranium and Plutonium Cases

As in the XRF cases, three simulations were completed to compare to measured data of a mixed uranium and plutonium solution. The mass ratios of uranium to plutonium ranged from 103:1 to 82:1.

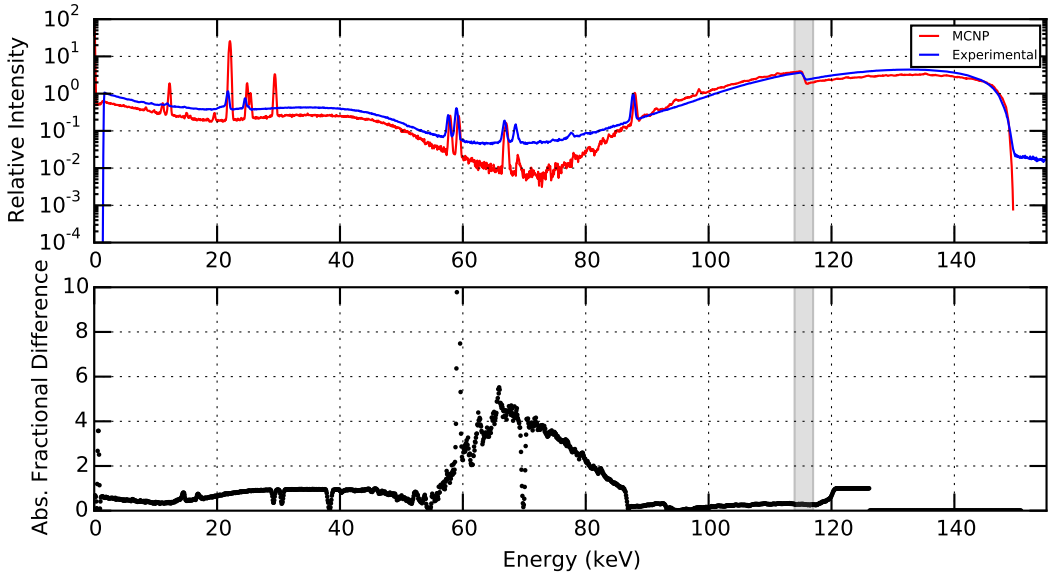


**Figure 6.31:** Pulse height spectrum of 107.5 $^{g}/_{L}$ uranium and plutonium at a mass ratio of 103:1 of uranium to plutonium. Deviation of the model from measured data remains relatively constant in energy regions of interest. Lower energies are more accurately represented between 42 and 59 keV.

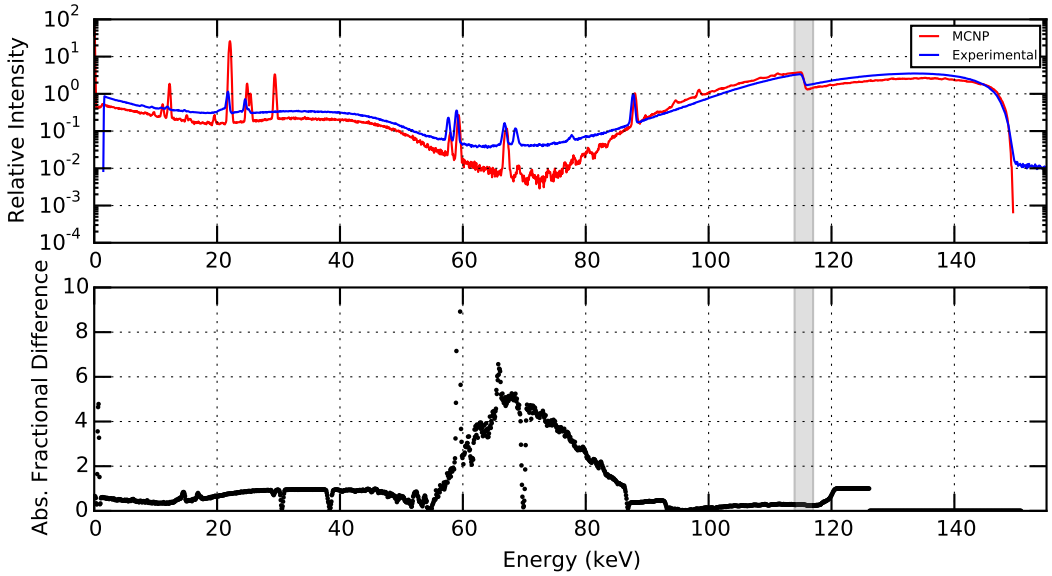


**Figure 6.32:** Pulse height spectrum of 160.8 $^{g}/_{L}$ uranium and plutonium at a mass ratio of 103:1 of uranium to plutonium. Deviation of the model from measured data remains relatively constant in energy regions of interest. Lower energies are more accurately represented between 42 and 59 keV.

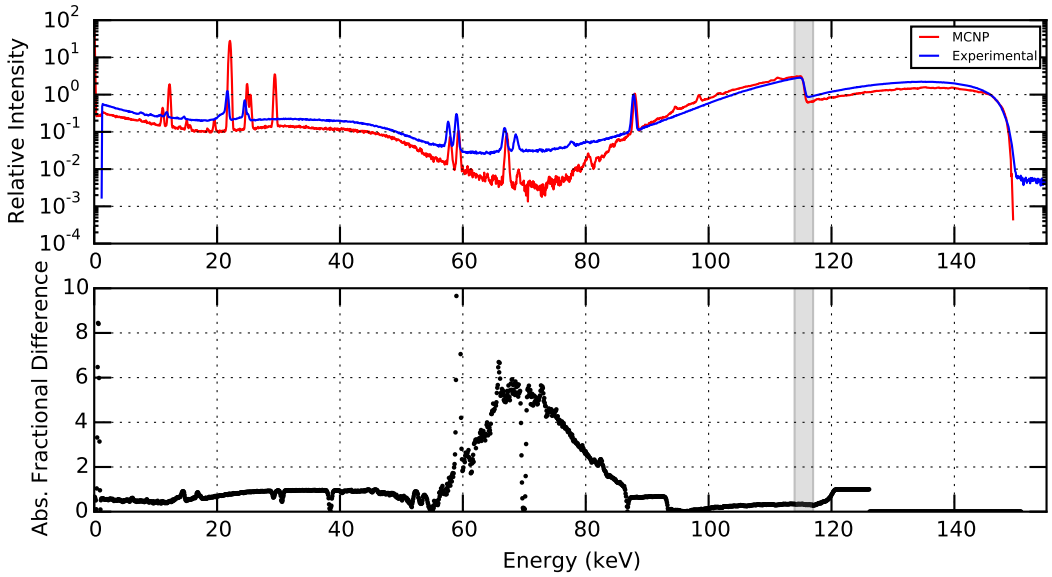


**Figure 6.33:** Pulse height spectrum of 243.3 $^{g}/_{L}$ uranium and plutonium at a mass ratio of 82.81:1 of uranium to plutonium. Deviation of the model from measured data remains relatively constant in energy regions of interest. Lower energies are more accurately represented between 42 and 59 keV.

The mixed actinide samples showed very similar characteristics to those of the uranium only samples. The depression of the spectrum around the tungsten K x-ray emissions were similar across all of the mixed actinide samples. The drop in the modeled spectrum in this region is likely due to an inconsistency in the x-ray source.

### 6.1.4 K-edge Continuum Analysis

A further analysis of the K-edge cases was performed to quantify the magnitude of the K-edge drop at the uranium K-edge. This was calculated by taking the continuum intensity (i.e. the area under the spectrum) for a region directly below and above the uranium K-edge energy of 115 keV. Analyzing the spectrum in this fashion will indicate whether the MCNP model of the K-edge drop is accurate with regard to the model.

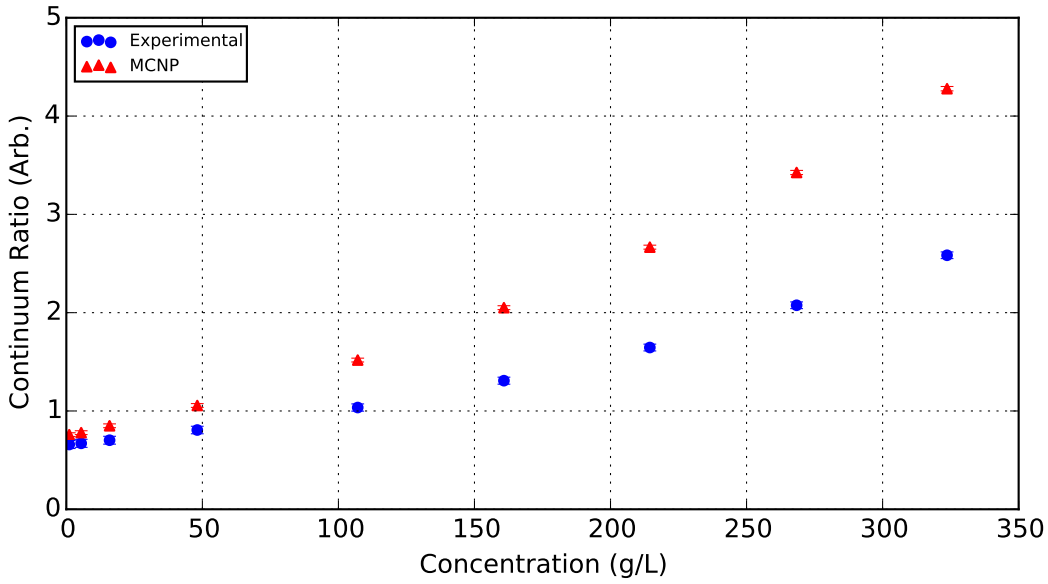


**Figure 6.34:** K-edge ratio drop as a function of energy for uranium only samples.

The K-edge drop of the model when compared to the measured data is fairly accurate below 50 $^{g}/_{L}$, however it begins to deviate above this concentration. This deviation increases as the concentration increases up to a maximum difference of 61% above the measured value at 323.7 $^{g}/_{L}$. This difference is attributed to an error in the MCNP photon library just as in the XRF simulations.

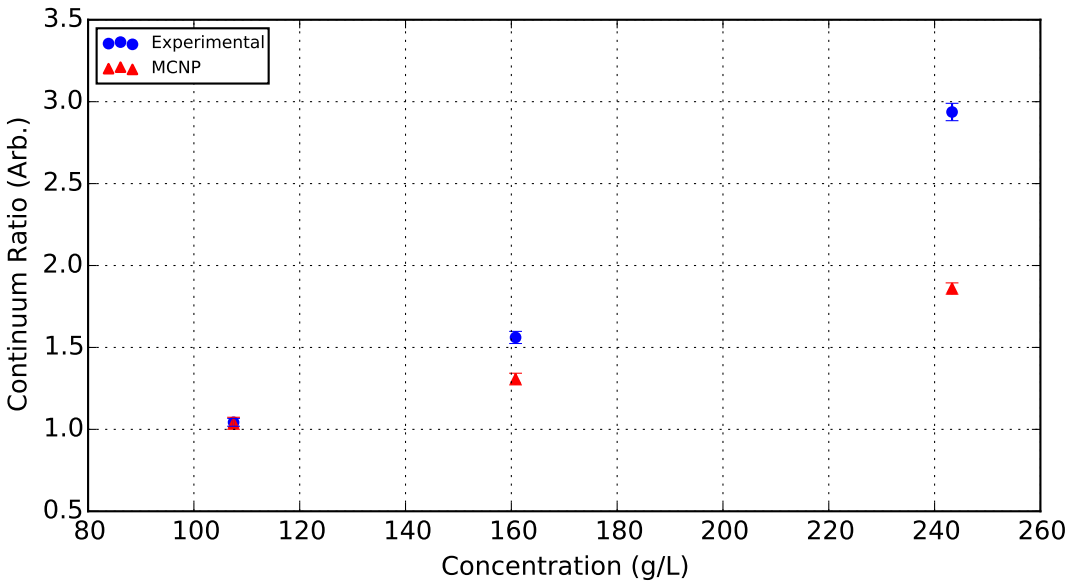


**Figure 6.35:** K-edge ratio drop as a function of energy for uranium and plutonium samples.

Just as in the the uranium only solutions the deviation between the measured and modeled data manifests itself above the 160.8 $^{g}/_{L}$ case. This increases to a maximum of 61% at the 243.4 $^{g}/_{L}$ case. Again, it is believed that this due to an inconsistency in the MCNP photon library.

A similar linear fit was performed on the K-edge transmission drop difference from the measured data. The difference in the modeled drop from the measured drop is shown in Figure 6.36

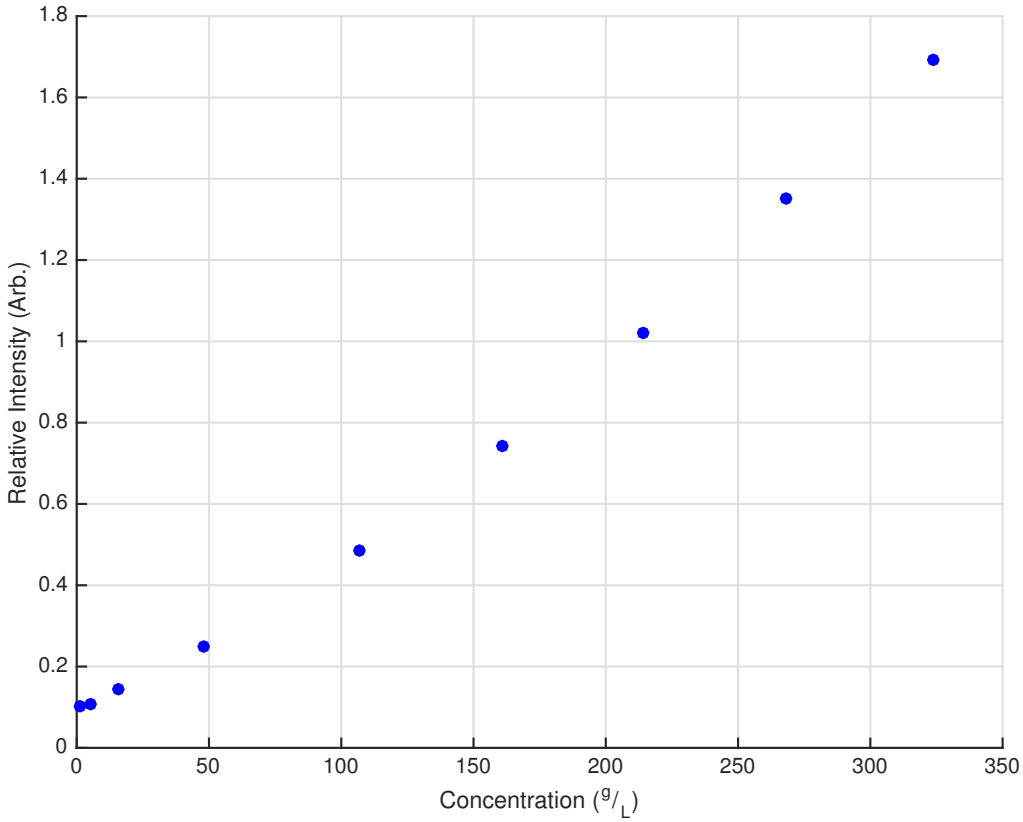


**Figure 6.36:** Difference in the modeled K-edge tansmission from from the measured data.

The linear fit was only performed for the uranium K-edge as the plutonium K-edge was not visible in each measurement due to the low concentration of plutonium when compared to uranium. The parameters for this fit are given in Table 6.2

**Table 6.2:** Linear fit parameters for the K-edge continuum drop difference for uranium.

| Element | $a_1$ | $a_2$ | $R^2$ |
|---|---|---|---|
| U | 0.004835 | 0.04006 | 0.9896 |

As in the fit analysis to the K x-ray peak intensities, the K-edge continuum drop difference has a roughly linear increase as the concentration of uranium increases.

## 6.2 X-ray Intensity and K-edge Response

A distinct deviation in both the K x-ray emissions and the K-edge continuum drop was identified in the analysis. The MCNP model over predicts the XRF response and the K-edge absorption leading to a response indicative of a higher concentration of actinide in solution. Further exploration of this was performed by building a simple beam and foil geometry in MCNP. An XY slice of the geometry is shown in Figure 6.37.

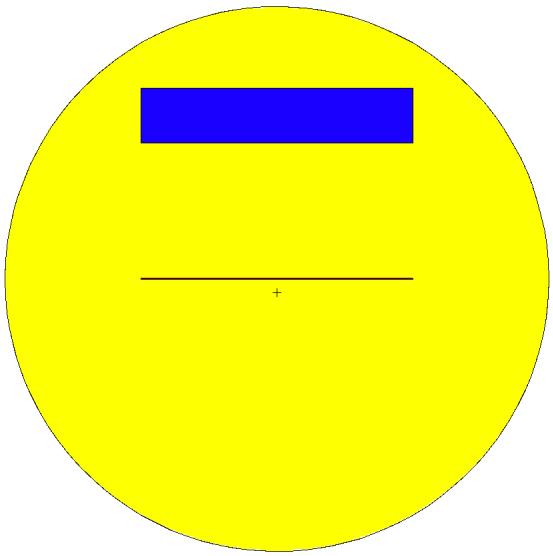

**Figure 6.37:** MCNP geometry of the K-edge check simulation. The source is placed at (0,-1,0) directly below the metal foil, and germanium detector.

The analysis was performed by choosing three different elements with K-edge energies different enough cover the energy range from 0 to 150 keV with a higher density. The elements chosen for the analysis were rhodium ($\rho = 12.45\ ^{g}/_{cm^3}$, z = 45), iridium ($\rho = 22.65\ ^{g}/_{cm^3}$, z = 77), and uranium ($\rho = 19.05\ ^{g}/_{cm^3}$, z = 92). The density of each sample was adjusted in MCNP along with calculating the realtive intensity drop and the values were compared. The MCNP intensity drop was determined by bombarding the foil with a flat x-ray spectrum bracketing the K-edge energy of each element. Figures 6.38 to 6.40 show the relative transmission to the fractional density of foils of rhodium, iridium, and uranium.

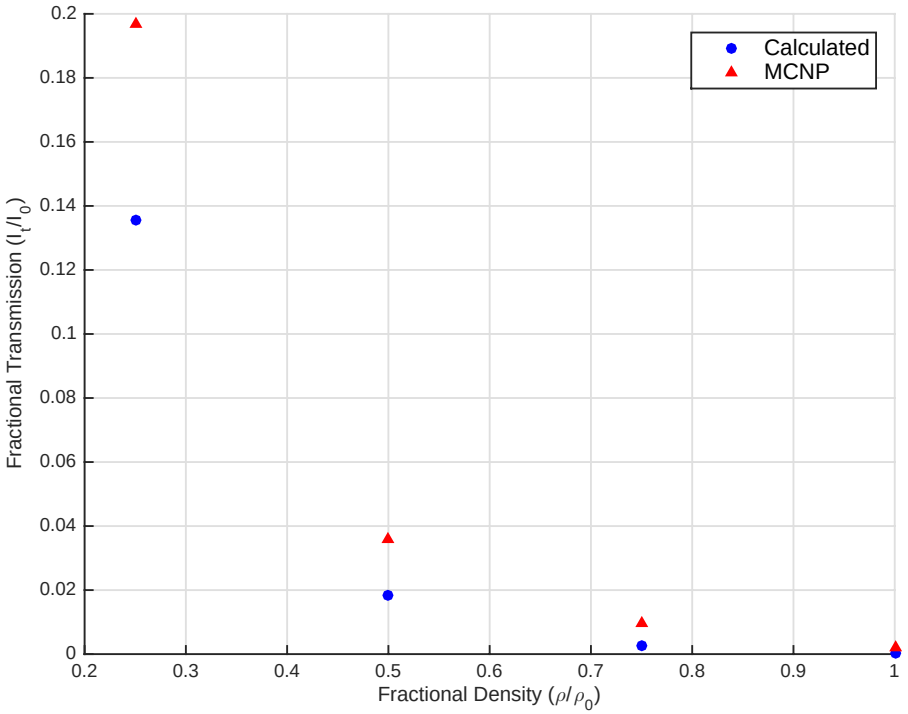


**Figure 6.38:** Fractional transmission of a flat x-ray spectrum as a function of the fractional density of rhodium. At lower fracitonal densities the MCNP model over estimates the transmission of x-rays.

As the fractional density of the rhodium foil increases the modeled values approach the calculated values. There is a large inconsistency at the lower fractional density.

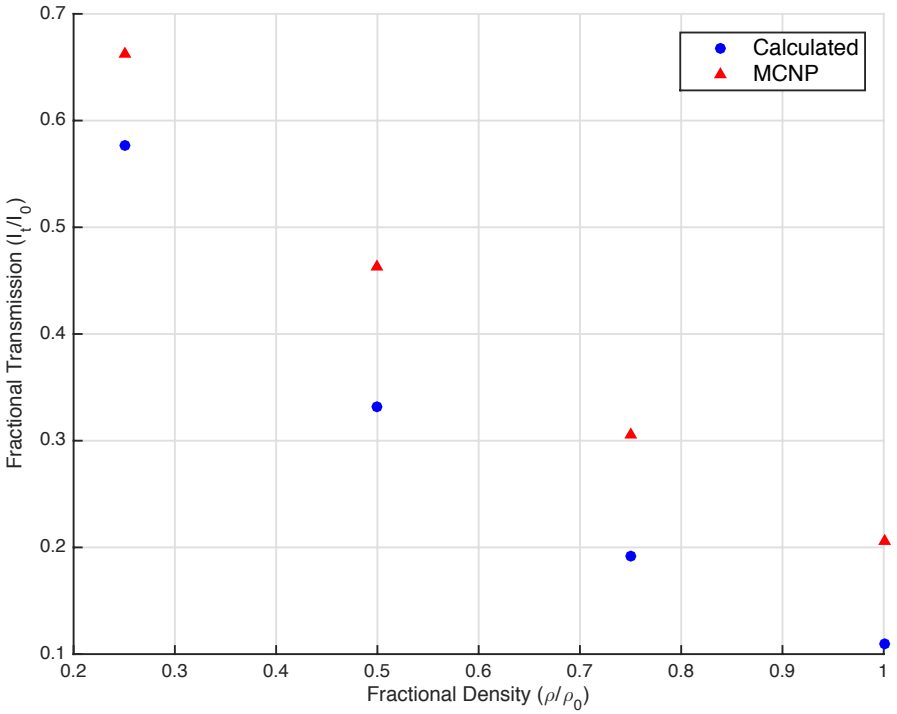


**Figure 6.39:** Fractional transmission of a flat x-ray spectrum as a function of the fractional density of iridium. A consistant difference between the modeled and calculated transmission was found at all fractional densities.

The modeled fractional transmission of photons in the iridium foil does not approach the calculated values for all fractional densities. There is a large deviation that remains constant for each density.

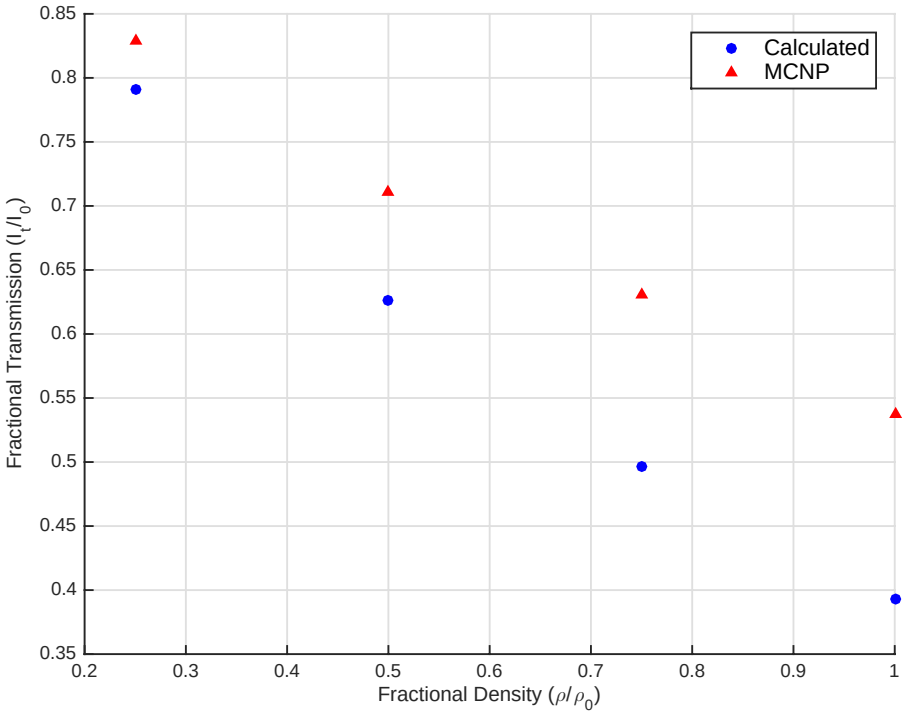


**Figure 6.40:** Fractional transmission of a flat x-ray spectrum as a function of the fractional density of uranium. The MCNP simulation increases in deviation from the calculated data as the fractional density of uranium increases.

As the fractional density of uranium in the sample increases the deviation of the modeled transmission increases. This confirms that the root cause of both the K-edge drop and the K x-ray emission overestimation appears to originate in the MCNP photon library. While this multi-element and multi-density study only begins to identify the issue in the photon libraries, it does indicate where the inconsistencies originate.

# Chapter 7

# Pyrochemical Cases

## 7.1 Molten Salt Solutions

A series of pyrochemical solutions were simulated using the HKED model upon completion of the validation studies. These solutions were modeled after salt taken from both the INL Mk.4 and Mk.5 electrorefiner. The Mk.4 electrorefiner was used to process blanket fuel from the EBR-II reactor and thus has higher concentrations of plutonium and neptunium. This electrorefiner also has a liquid cadmium cathode to capture plutonium and other minor actinides from the electrolyte. The Mk.5 electrorefiner processed driver fuel and thus contains salt with a higher uranium content (30).

### 7.1.1 X-ray Fluorescence

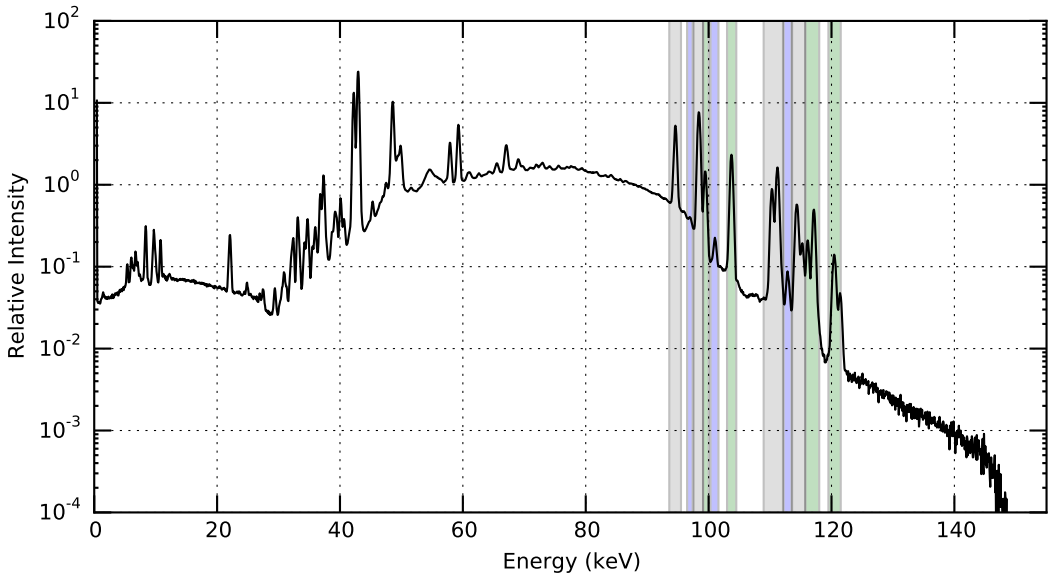


**Figure 7.1:** XRF pulse height spectrum of salt taken from the INL Mk.4 electrorefiner. This electrorefiner processed blanket fuel from EBR-II. Several prominent peaks of uranium, plutonium, and neptunium are visible.

The sample of Mk.4 electrorefiner salt shows a large variety of x-ray peaks. The most prominent peaks are uranium $K_\alpha$ emissions at 84.6 and 98.4 keV, respectively. Plutonium $K_\alpha$ peaks are visible at 99.5 and 103.7 keV for the $K_\alpha$ and $K_\beta$ emissions. Neptunium $K_\alpha$ peaks are visible at 97.1 keV and 101.6 keV. The $K_\beta$ peaks of uranium, plutonium, and neptunium are grouped tightly together between 110 and 123 keV.

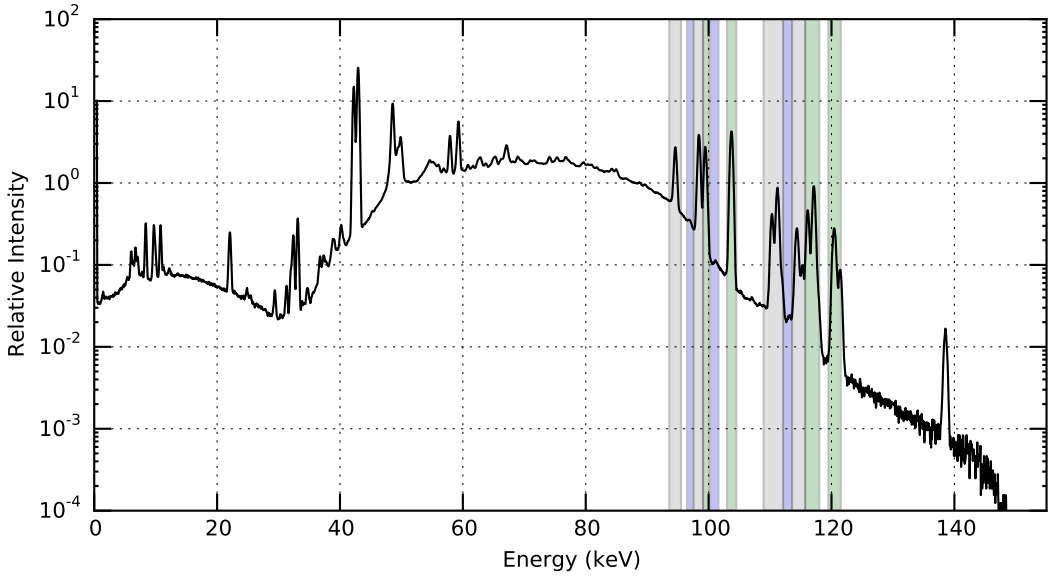


**Figure 7.2:** XRF pulse height spectrum of salt taken from the INL Mk.5 electrorefiner. This electrorefiner processed blanket fuel from EBR-II. Several prominent peaks of uranium, plutonium, and neptunium are visible.

The peaks shown in Figure 7.2 are similar to those presented in the aqueous simulations with the exception of the plutonium emission lines. These peaks are found at 103.7 and 117.3 keV for $K_{\alpha 1}$ and $K_{\beta 1}$ emissions, respectively. A smaller peak at 99.5 keV corresponding to the plutonium $K_{\alpha 2}$ emission is shown directly beside the uranium $K_{\alpha 1}$. Neptunium $K_{\alpha}$ peaks are slightly visible at 97.1 keV and 101.6 keV, however they are not at prominent as in the Mk.4 case shown in Figure 7.1.

### 7.1.2 K-edge Transmission

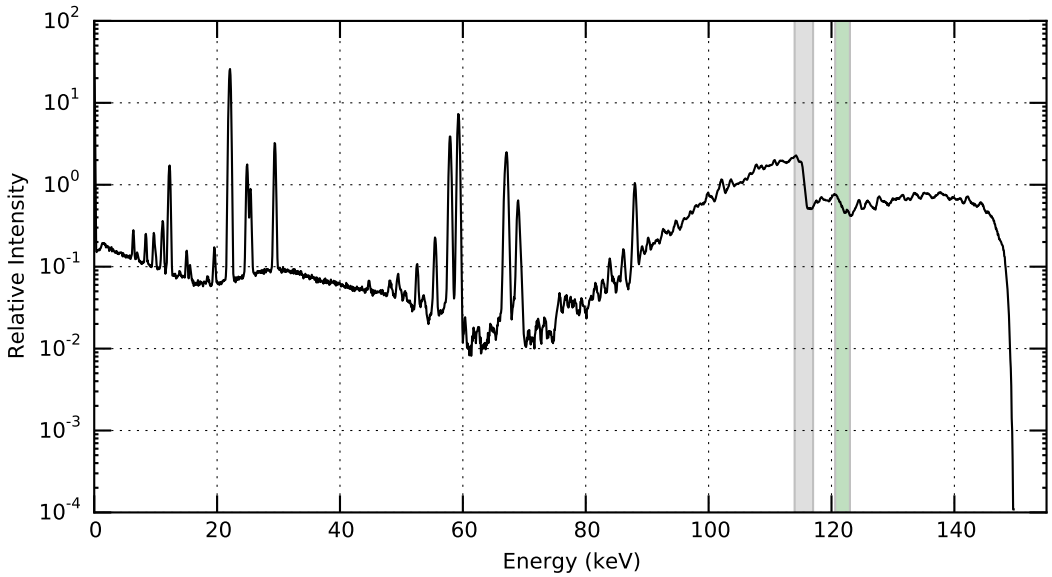


**Figure 7.3:** KED pulse height spectrum of salt taken from the INL Mk.4 electrorefiner. Two distinct K-edge drops can be seen between 115 and 122 keV.

The Mk.4 electrorefiner K-edge simulation shows two distinct K-edge drops for both uranium and plutonium. The uranium to plutonium mass ratio for this case was 2:1. Neptunium and americium are also present in solution, however the concentration is not high enough to produce another K-edge drop. In both simulations several other XRF peaks can be seen. There are primarily from tungsten between 55 and 70 keV. The $^{109}$Cd 88 keV peak is prominent along with the accompanying x-rays at lower energies.

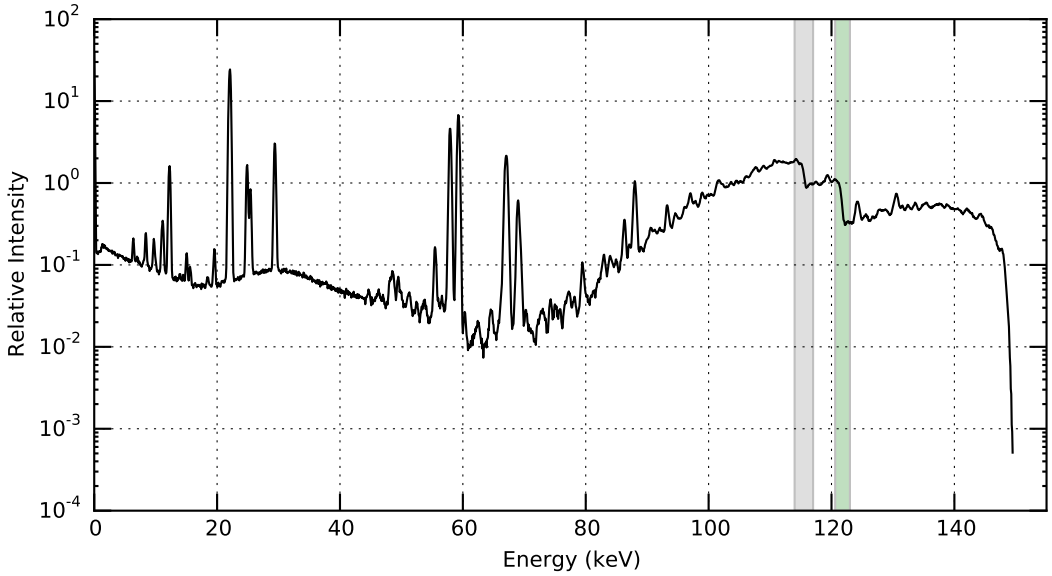


**Figure 7.4:** KED pulse height spectrum of salt taken from the INL Mk.5 electrorefiner. This electrorefiner processed blanket fuel from EBR-II. Uranium and plutonium K-edge drops are prominent between 115 and 121 keV.

Figure 7.4 shows the pulse height spectrum of a K-edge transmission simulation of a similar U:Pu salt solution as before. The uranium K-edge is readily apparent at 115.6 keV. The plutonium K-edge (expected at 121.8 keV) is not readily seen due the the relatively low concentration of plutonium in solution.

While modeling novel salt solutions of just uranium and plutonium provides insight into how the system will respond to samples taken from an electrorefiner, more realistic studies are needed to predict the system response to more realistic solutions.

## 7.2 Dense Pyrochemical Samples

Modeling of the liquid cathode in an electrorefiner is important, from a safeguards perspective, due to the presence of plutonium. During the electrorefining process plutonium, along with other minor actinides, will transport through the salt to the

liquid cathode at the bottom of the electrorefiner. Measuring the plutonium content of a liquid cathode presents issues such as higher temperatures, toxic metals, higher radiation fields, and a higher density than aqueous or pyrochemical salt solutions.

One solution to the issue of higher density was to decrease the overall thickness of the sample. The sample thickness was set such that the photon transmission characteristics are similar to the aqueous solutions. This was accomplished by determining the relative attenuation of a 323.3 $^{g}/_{L}$ aqueous sample then using that relative attenuation, denoted as $I_{aq}$, to determine the equivalent transmission thickness in a denser sample.

$$t = -\left(\frac{1}{\sum_i \mu_i}\right) ln\left(\frac{I_t}{I_0}\right) \tag{7.1}$$

The equivalent transmission thicknesses for liquid cathodes containing bismuth and cadmium and the overall attenuation for both are shown in Table 7.1.

**Table 7.1:** Pyrochemical sample attenuation coefficients and thicknesses.

| **Sample** | **Attn. Coeff. ($cm^{-1}$)** | **New Thickness (cm)** |
|---|---|---|
| Voloxidation Powder | 32.83 | 0.27 |
| Liquid Cadmium Cathode | 10.80 | 0.82 |
| Liquid Bismuth Cathode | 46.51 | 0.19 |
| Metallic Foil | 37.22 | 0.24 |

### 7.2.1 X-ray Fluorescence

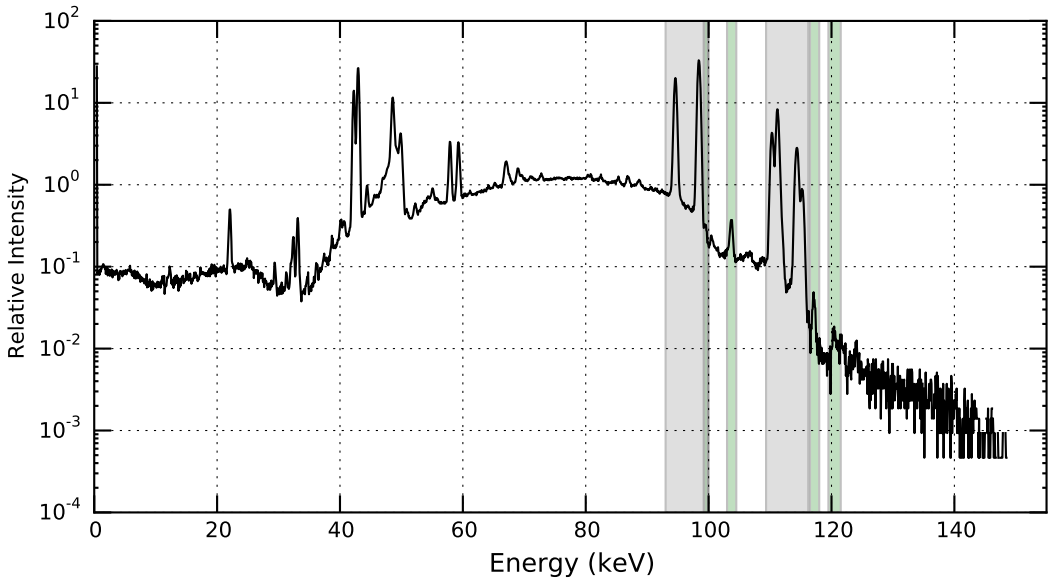


**Figure 7.5:** XRF pulse height spectrum of a voloxidation powder sample. The average sample density is 16.81 $^{g}/_{cm^3}$.

The primary actinide component of the voloxidation powder has several prominent K x-ray emissions corresponding to uranium and plutonium. The plutonium peaks are faint when compared to the uranium emissions, however the plutonium $K_{\alpha 1}$ peak is seen at 103 keV. The plutonium $K_{\beta}$ are visible but their resolution is poor. Uranium $K_{\alpha}$ and $K_{\beta}$ peaks are clearly visible.

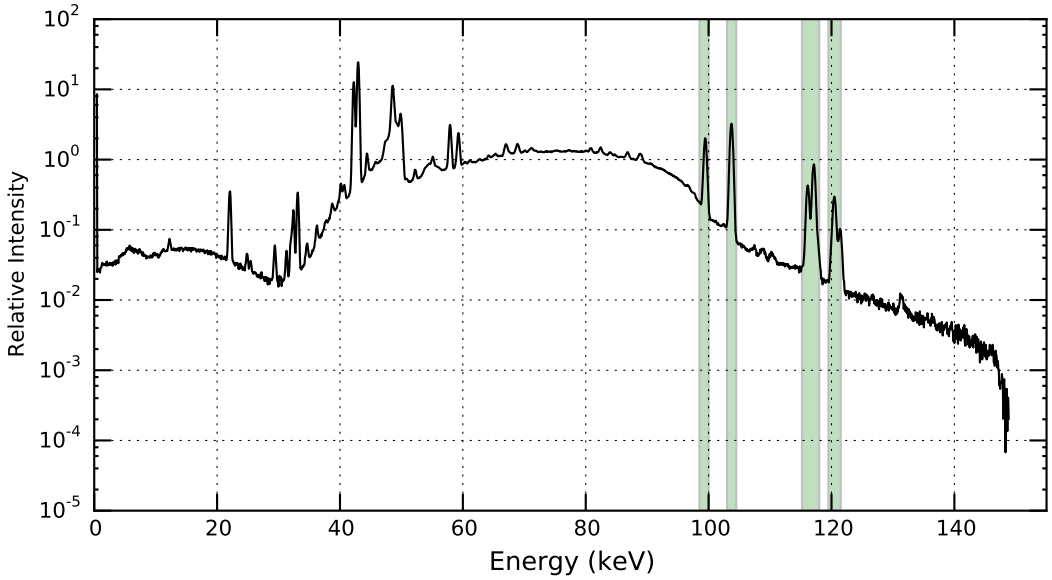


**Figure 7.6:** XRF pulse height spectrum of a liquid cathode containing cadmium and plutonium 0.82 cm thick with a density of 10.97 $^{g}/_{cm^3}$.

Tracking the plutonium in the liquid cadmium cathode is quite important from a safeguards perspective. A slight alteration to the sample geometry is all that is required to facilitate a measurement of the plutonium content. Prominent plutonium $K_{\alpha}$peaks are visible at 99.5 keV and 103.7 keV. The associated $K_{\beta}$ peaks are also shown between 115 and 123 keV. Cadmium K x-ray peaks are seen between 20 and 40 keV along with gadolinium peaks from the filter at the end of the collimator.

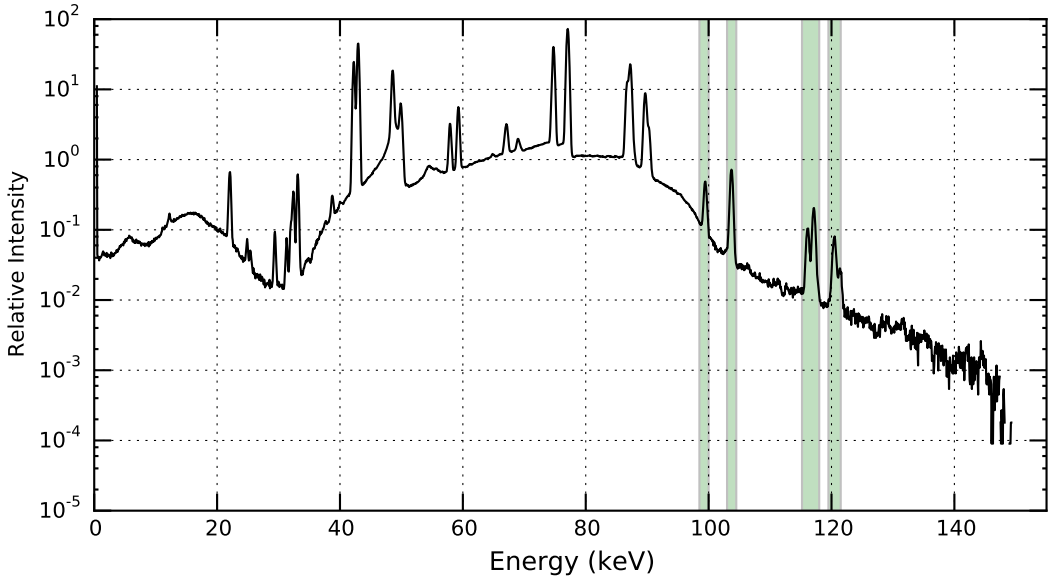


**Figure 7.7:** XRF pulse height spectrum of a liquid cathode containing bismuth and plutonium 0.19 cm thick . Plutonium K x-ray emissions are shown in green.

Bismuth is another candidate for the liquid cathode due to its lower toxicity compared to cadmium. A simulation was ran to determine what the XRF response would be to a similar plutonium loading as in the cadmium cathode case. As in the cadmium cathode XRF spectrum clear plutonium peaks are visible in addition to K x-ray peaks of bismuth cathode at 77 and 74 keV for $K_{\alpha 1}$ and $K_{\alpha 2}$, respectively.

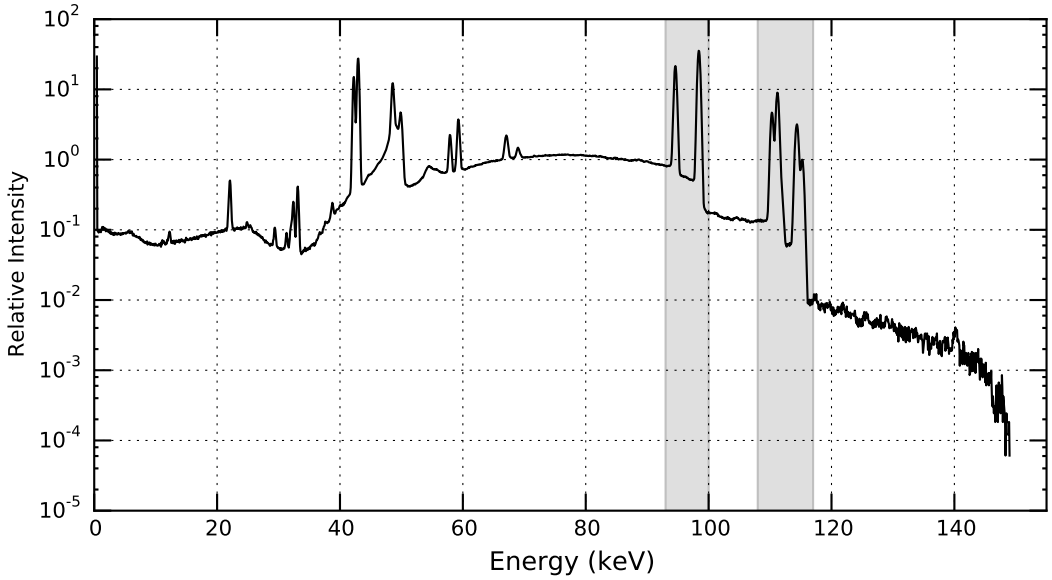


**Figure 7.8:** XRF pulse height spectrum of a metallic foil of uranium 0.24 cm thick with a density of 19.05 $^{g}/_{cm^3}$. X-ray emissions of uranium fall within the grey ROIs.

The densest sample to be measured was a foil of uranium metal. The density of the sample was 19.1 $^{g}/_{cm^3}$. Very prominent XRF peaks are visible between 94 and 115 keV for the $K_\alpha$ and $K_\beta$ emissions, respectively. These results indicate that it is possible to assay the metallic product from a pyroprocessing operation using a XRF measurement.

### 7.2.2 K-edge Transmission

As in the XRF measurements, a series of simulations were completed using the denser samples for the K-edge measurement.

The voloxidation powder KED response does not clearly show a uranium K-edge at 115.7 keV. There is a noticeable drop in the relative intensity at that energy, however the K-edge is not clearly defined.

As in the voloxidation powder simulation, the liquid cadmium cathode KED measurement does not appear to show any plutonium K-edge at 121.8 keV. A lack

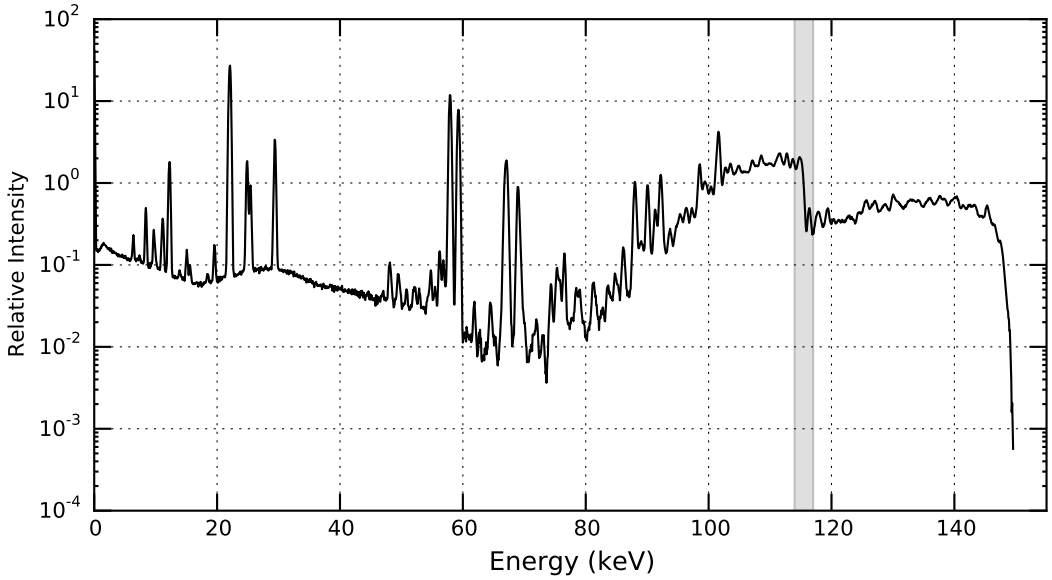


**Figure 7.9:** KED pulse height spectrum of a voloxidation powder sample 0.27 cm thick. A rough uranium (the primary component of the sample) K-edge is visible at 115.7 keV.

of resolution at the upper energies indicates that more histories may be required to resolve this part of the spectrum.

Figure 7.11 shows the KED response for a sample loaded with bismuth and plutonium. The bismuth K-edge at 83 keV results in a substantial x-ray transmission drop that affects the higher energies. This obscures the uranium K-edge at 115 keV. While the bismuth cathode may be favorable due to a lower toxicity, it does not appear to have favorable measurement properties.

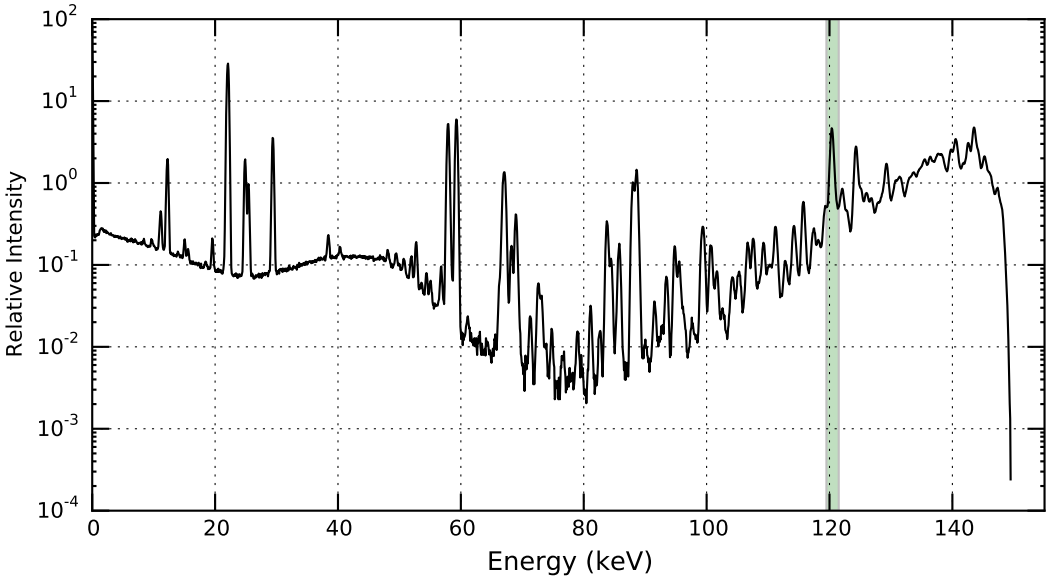


**Figure 7.10:** KED pulse height spectrum of a liquid cadmium cathode 0.82 cm thick.

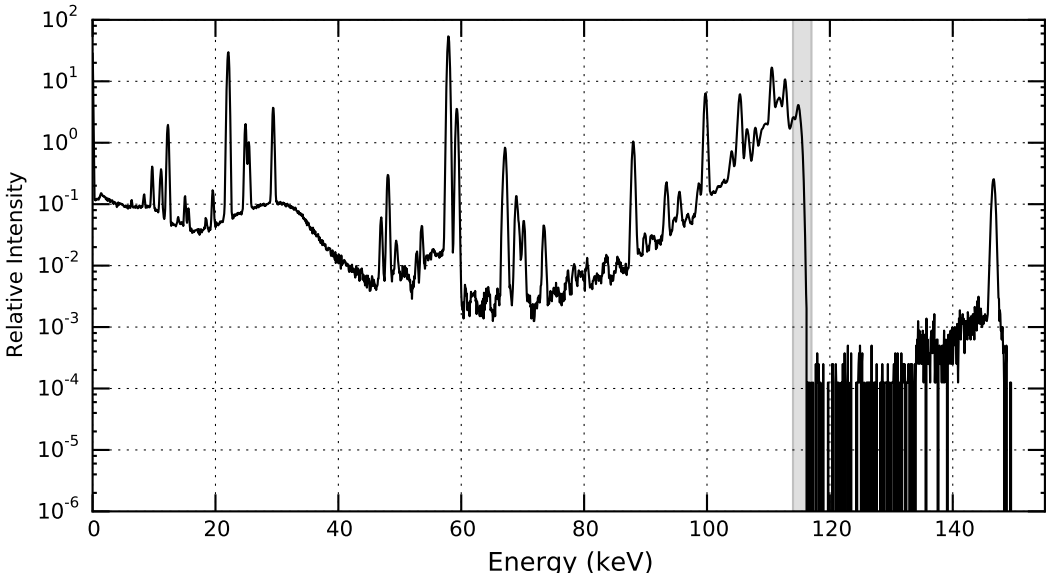


**Figure 7.12:** KED pulse height spectrum of a metallic foil of uranium 0.24 cm thick. The area around the uranium K-edge is denoted in grey.

The K-edge drop shown in Figure 7.12 does occur at the correct energy (115.7 keV) however the substantial shielding effect of the sample does not provide a clear

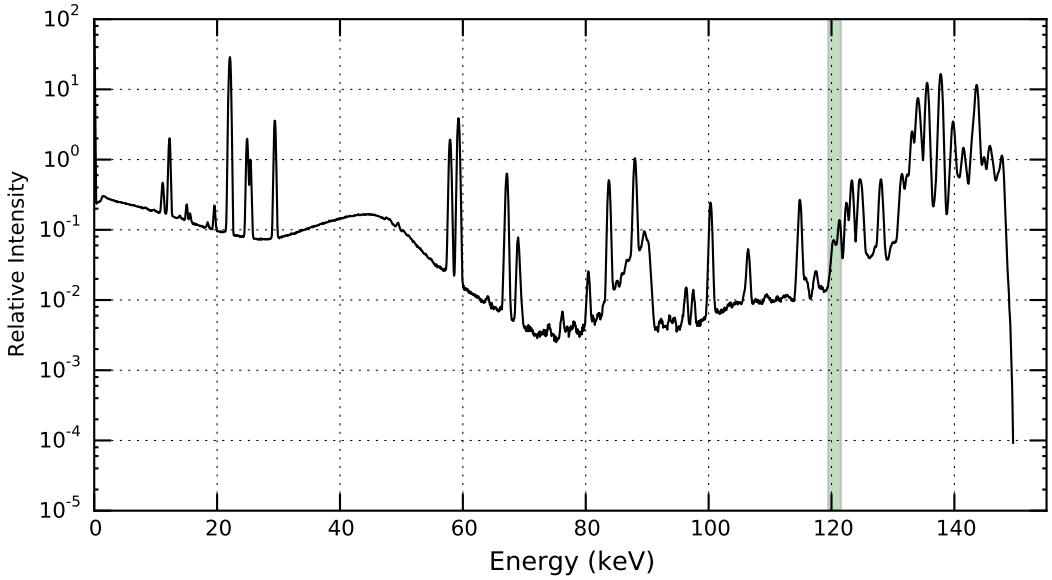


**Figure 7.11:** Liquid cathode containing bismuth and plutonium 0.19 cm thick. The green ROI brackets the plutonium K-edge at 121.8 keV.

upper energy component of the continuum. The lack of resolution at higher energies indicates that this measurement would be difficult for a metallic sample of this thickness.

Each of the samples simulated do not currently have a method of assay other than destructive analysis. While the XRF measurements show promise to assay the more complex samples from a pyroprocessing operation, the KED measurements appear to have limited applicability due to the higher density of the sample. However, a simple modification to the HKED instrument, changing the geometry of the sample, results in an extended capability to measure samples in the instrument that would not otherwise be possible.

## 7.3 Surrogate Samples

One method proposed to lower the overall cost of subsequent scoping studies for HKED application to pyroprocessing is to use surrogates in place of plutonium. Uranium samples are generally easier to work with in a laboratory environment,

however plutonium containing solutions are more difficult to procure and work with from a logistics standpoint.

### 7.3.1 X-ray Fluorescence

Simulations of the XRF response to the thorium surrogates sought to mimic the concentrations found in the mixed uranium plutonium studies. The goal with the XRF response simulations is to show that the intensity and position of the thorium peaks in relation to the uranium peaks will mimic that of plutonium to uranium.

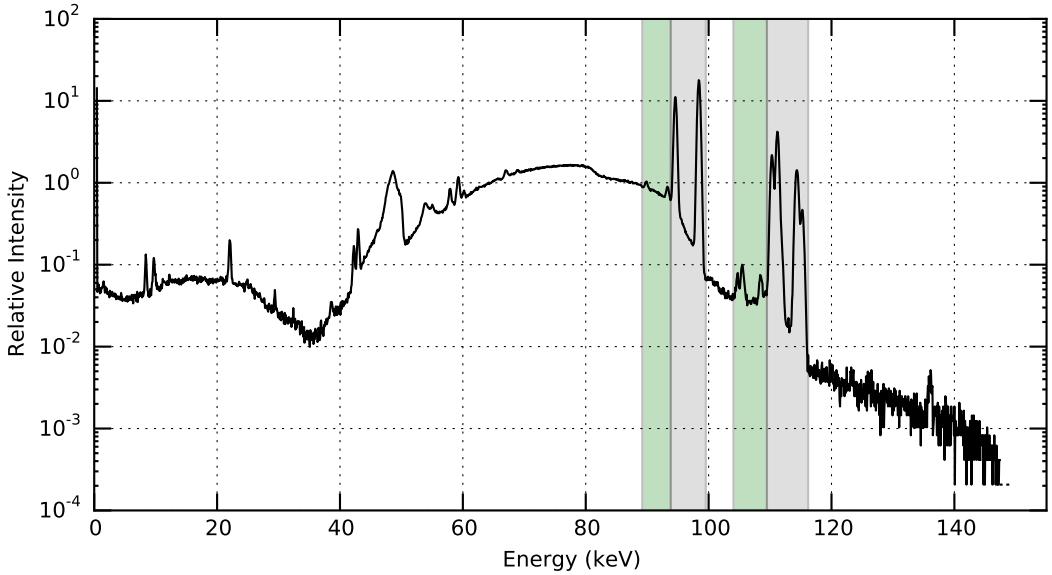


**Figure 7.13:** XRF pulse height spectrum of 107.5 g/L U at 103:1 U:Th mass ratio. Thorium x-ray emissions are denoted by the green ROI and uranium in grey.

Small thorium $K_{\alpha}$ peaks are visible at energies directly below the uranium K$\alpha$ peaks. Thorium $K_{\beta}$ peaks are also visible between the groups of uranium $K_{\alpha}$ and $K_{\beta}$ emissions.

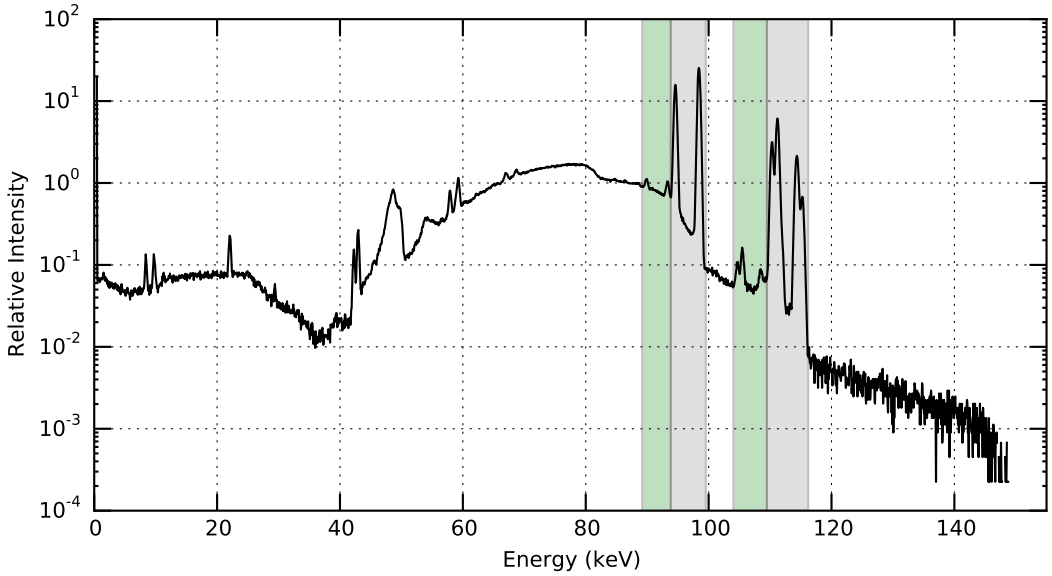


**Figure 7.14:** XRF pulse height spectrum of 160.8 g/L U at 103:1 U:Th mass ratio. Thorium x-ray emissions are denoted by the green ROI and uranium in grey.

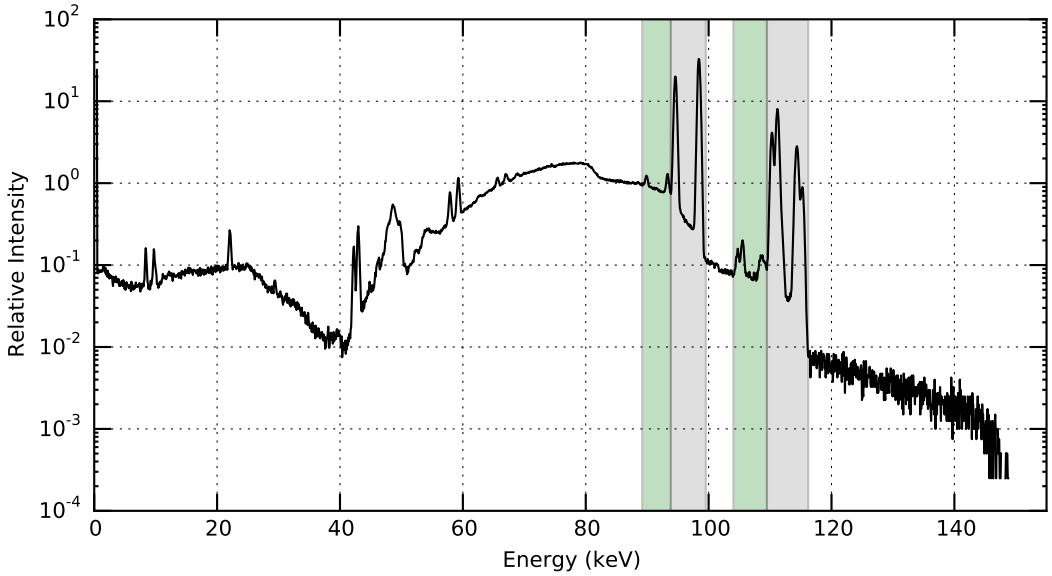


**Figure 7.15:** XRF pulse height spectrum of 213.0 g/L U at 102.4:1 U:Th mass ratio. Thorium x-ray emissions are denoted by the green ROI and uranium in grey.

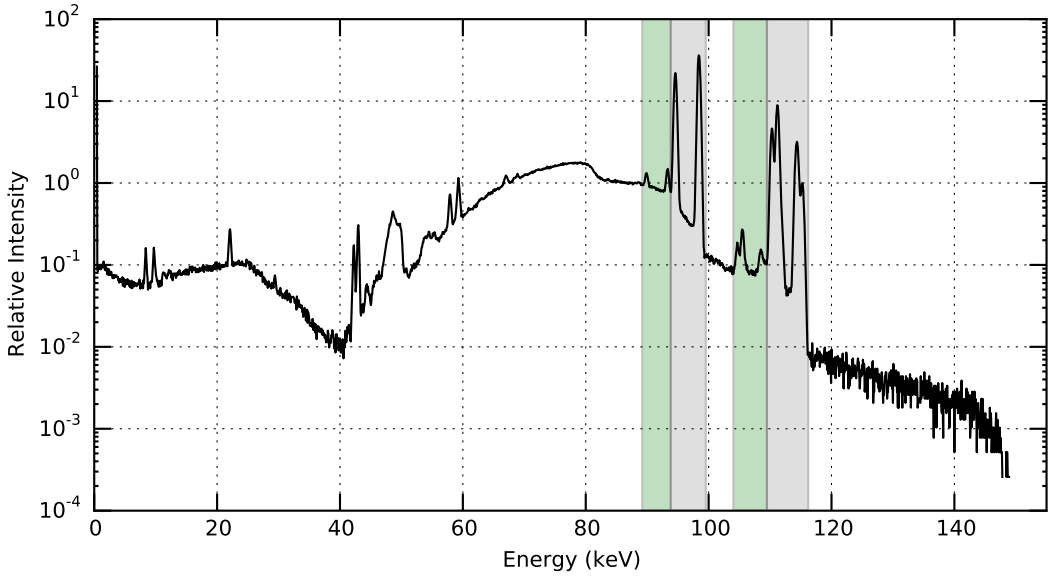


**Figure 7.16:** XRF pulse height spectrum of 243.3 g/L U at 82.81:1 U:Th mass ratio. Thorium x-ray emissions are denoted by the green ROI and uranium in grey.

Figures 7.13 to 7.16 depict the XRF detector response for increasing concentrations of uranium and thorium. The thorium $K_\alpha$ peaks are visible between 89 and 93 keV and $K_\beta$ peaks at 104 to 108 keV. While these peaks are at lower energies than those of plutonium they do lie at relatively the same proximity as plutonium K x-ray emissions.

In addition to the previous surrogate simulations, a series of simulations were completed with varied mass ratios of uranium to thorium. Samples were created with 234.3 $^{g}/_{L}$ concentration of uranium and varied mass ratios of 100:1, 20:1, 4:1, 2:1, and 1:1. Figures 7.17 to 7.21 depict the pulse height spectrum response for these samples.

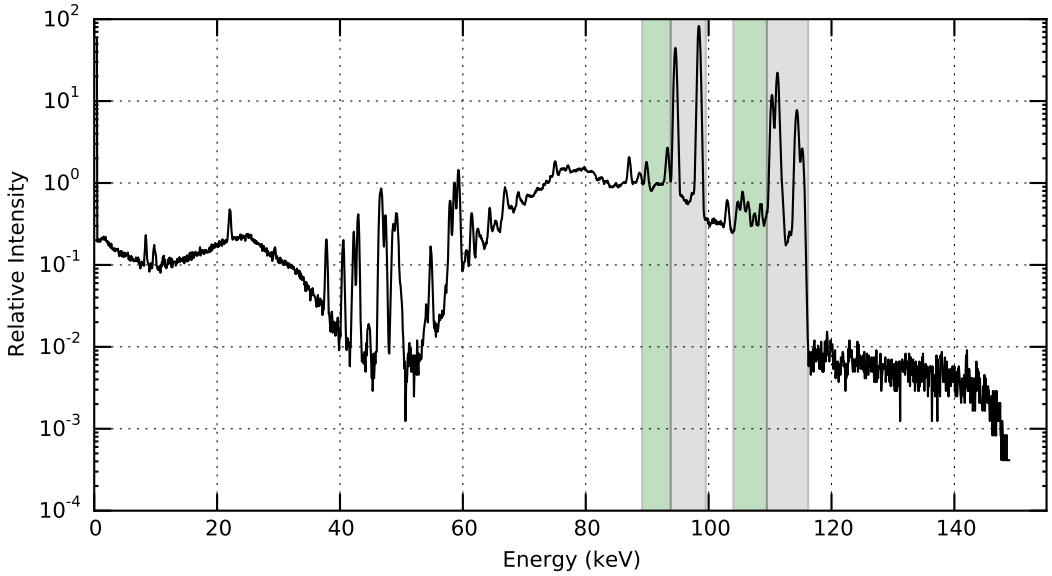


**Figure 7.17:** XRF pulse height spectrum of 243.3 g/L U at 100:1 U:Th mass ratio. Thorium x-ray emissions are denoted by the green ROI and uranium in grey.

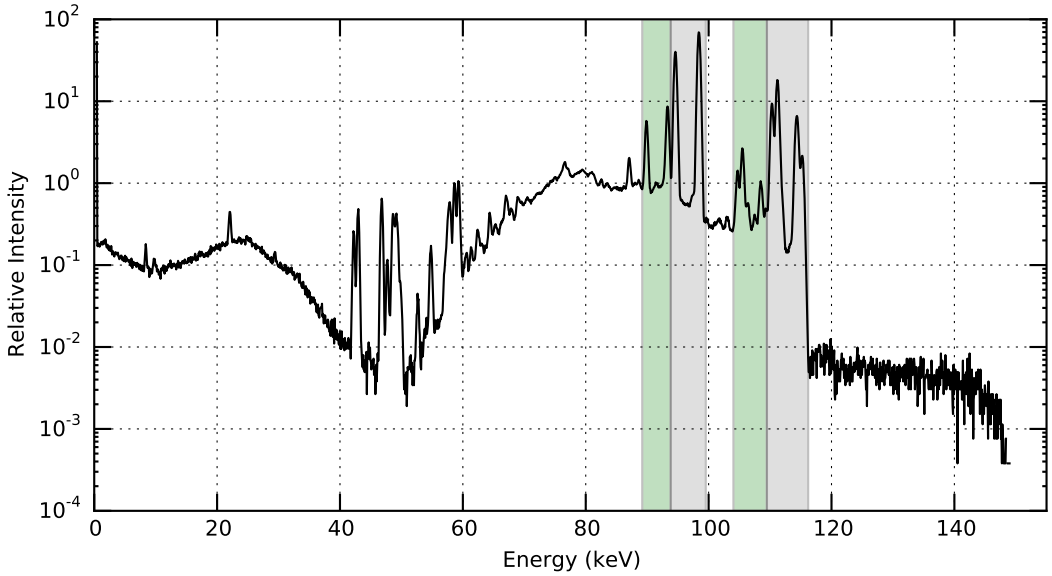


**Figure 7.18:** XRF pulse height spectrum of 243.3 g/L U at 20:1 U:Th mass ratio. Thorium x-ray emissions are denoted by the green ROI and uranium in grey.

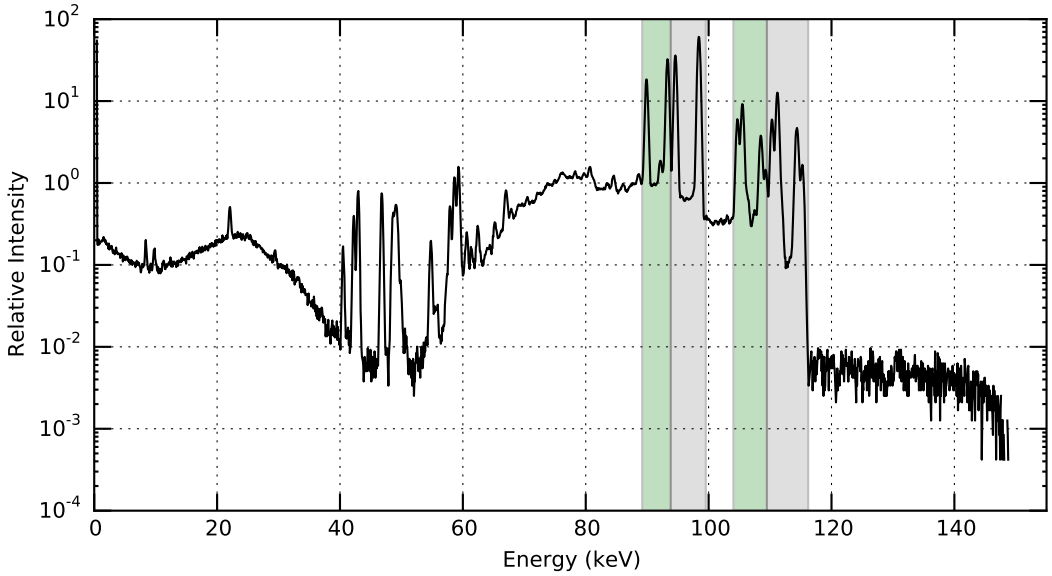


**Figure 7.19:** XRF pulse height spectrum of 243.3 g/L U at 4:1 U:Th mass ratio. Thorium x-ray emissions are denoted by the green ROI and uranium in grey.

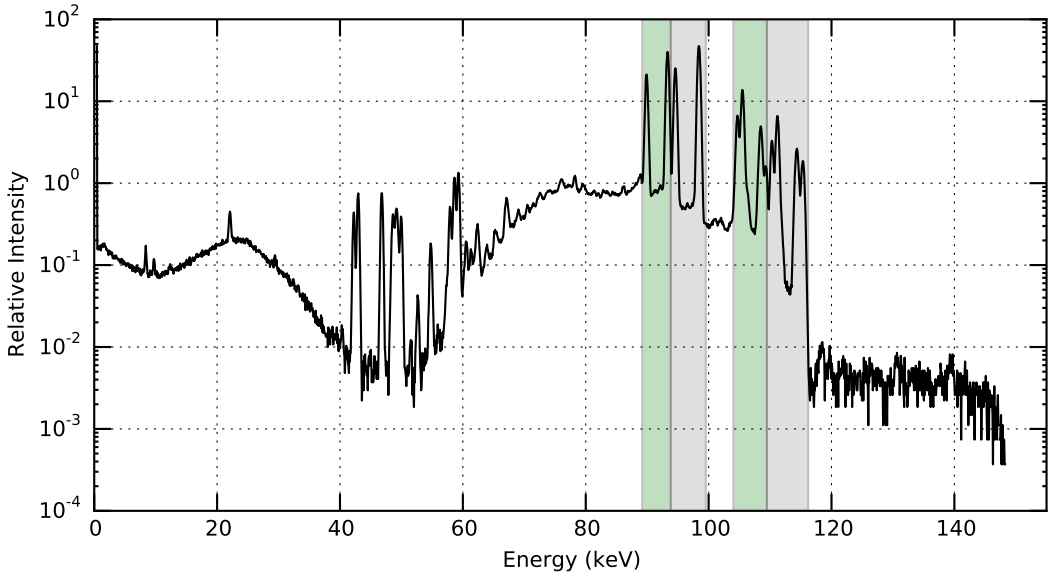


**Figure 7.20:** XRF pulse height spectrum of 243.3 g/L U at 2:1 U:Th mass ratio. Thorium x-ray emissions are denoted by the green ROI and uranium in grey.

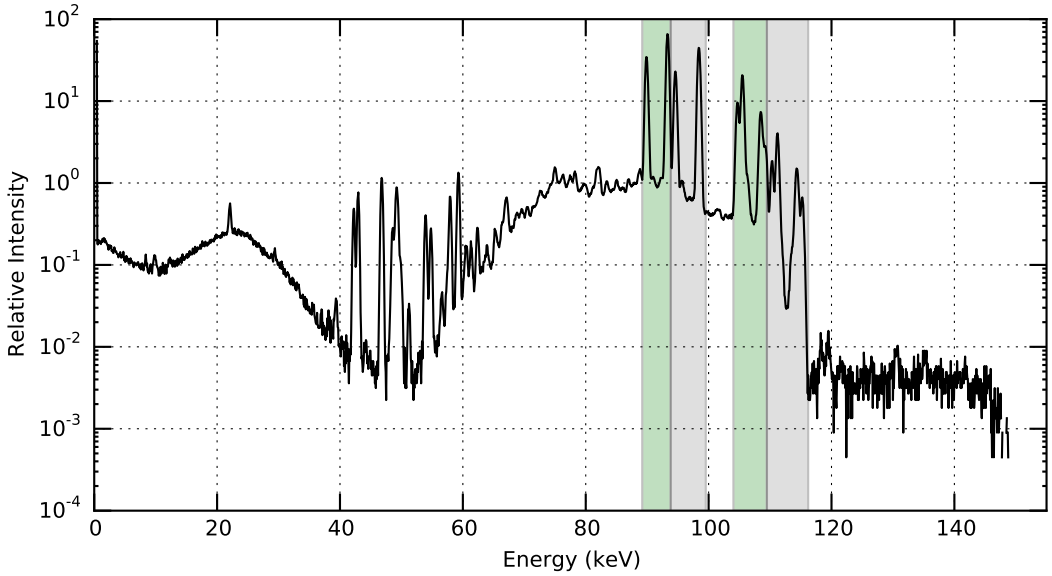


**Figure 7.21:** XRF pulse height spectrum of 243.3 g/L U at 1:1 U:Th mass ratio. Thorium x-ray emissions are denoted by the green ROI and uranium in grey.

As the concentration of thorium increases the intensities of each thorium K x-ray emission increases. These increases become as prevalent as the uranium emissions in the higher concentrations of thorium. The proximity of the peaks to each other, while at different energies than the plutonium peaks, do provide the similar intensity as the plutonium K x-ray emissions. The XRF emission characteristics shown for the mixed thorium and uranium samples indicate that thorium is indeed an acceptable surrogate for plutonium. K x-ray emission peaks for thorium are distinguishable from the uranium peaks to be independently quantified apart from other peaks in the spectra.

### 7.3.2 K-edge Transmission

Simulations were performed to determine the K-edge detector response to mixtures of thorium and uranium to determine if the response would be similar to that of the mixed plutonium and uranium samples. Sample compositions were chosen to mimic

those of the mixed uranium and plutonium cases completed in the model validation cases.

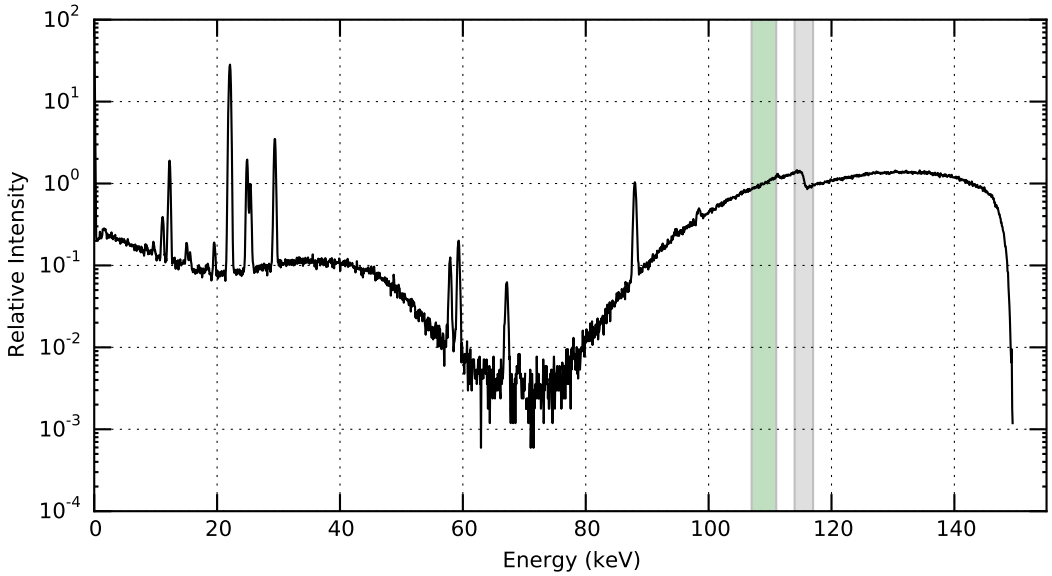


**Figure 7.22:** KED pulse height spectrum of 107.5 g/L U at 103:1 U:Th mass ratio.

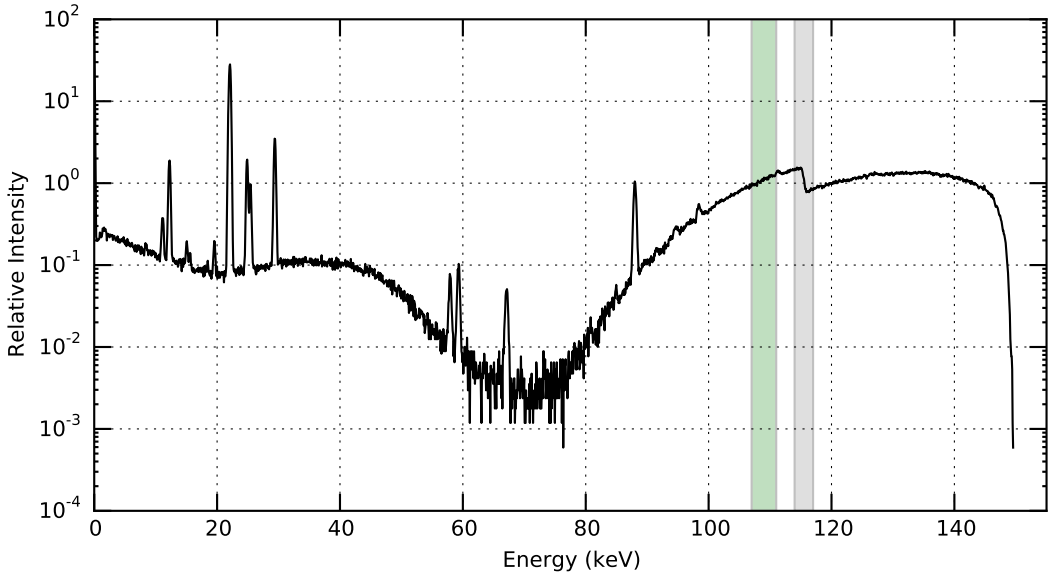


**Figure 7.23:** KED pulse height spectrum of 160.8 g/L U at 103:1 U:Th mass ratio.

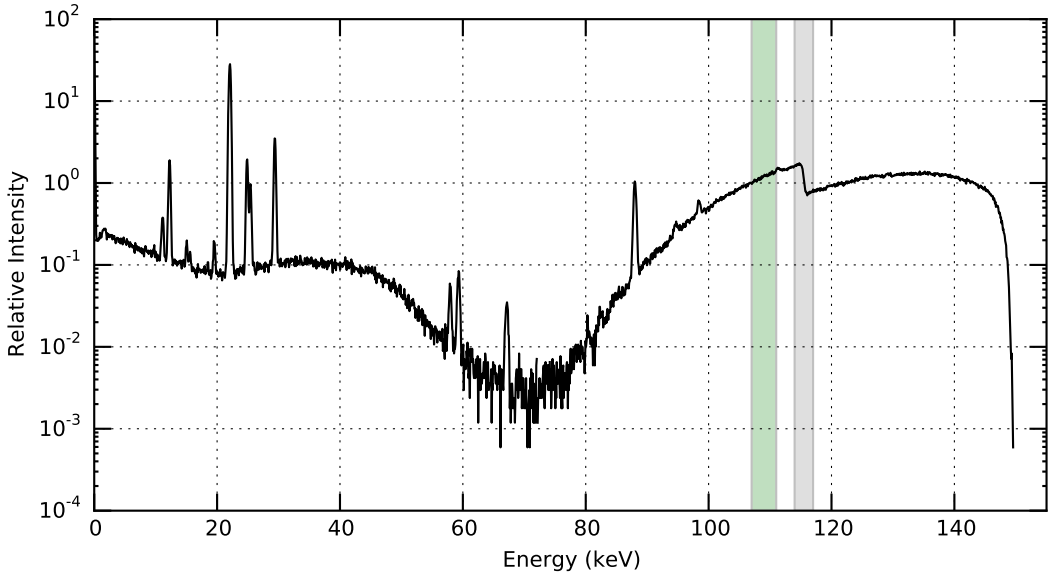


**Figure 7.24:** KED pulse height spectrum of 213.0 g/L U at 102.4:1 U:Th mass ratio.

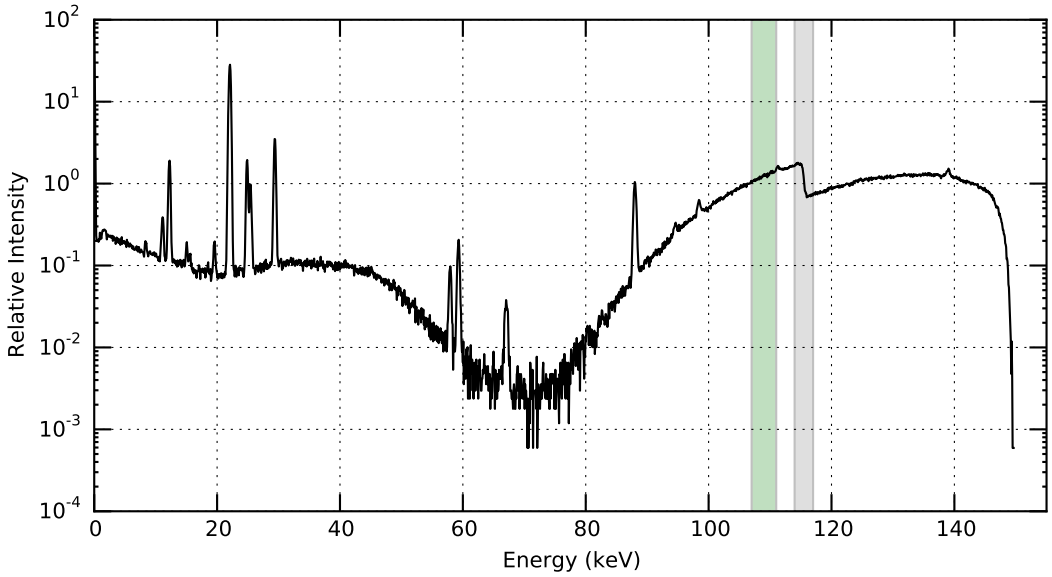


**Figure 7.25:** KED pulse height spectrum of 243.3 g/L U at 82.81:1 U:Th mass ratio.

Figures 7.22 to 7.25 show that the K-edge drop does indeed behave similarly as the mixed actinide uranium-plutonium samples. The continuum drop does increase as expected and the thorium K-edge at 109.6 keV is not visible in any spectrum, as in the uranium-plutonium simulations.

A series of simulations of varied uranium to thorium mass ratio were completed in addition to simulating a series of mixed uranium and plutonium samples matching the mass ratios of the mixed uranium and plutonium samples.

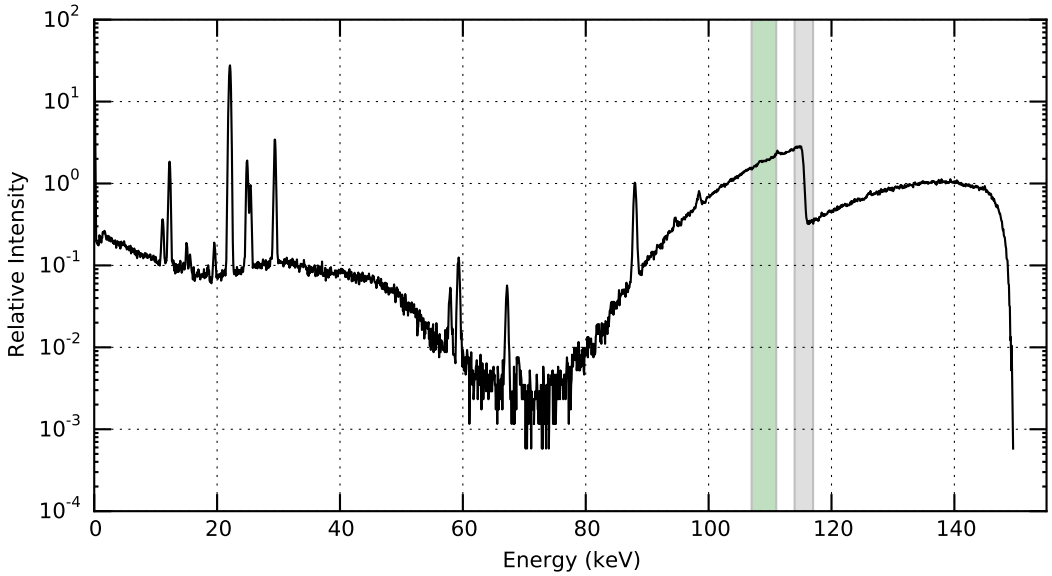


**Figure 7.26:** KED pulse height spectrum of 243.3 g/L U at 100:1 U:Th mass ratio.

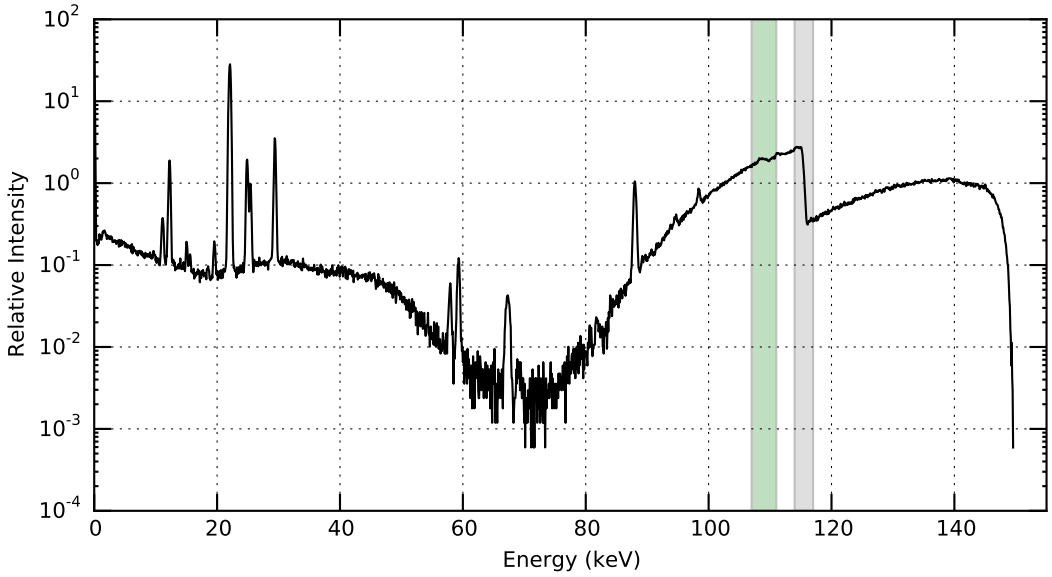


**Figure 7.27:** KED pulse height spectrum of 243.3 g/L U at 20:1 U:Th mass ratio.

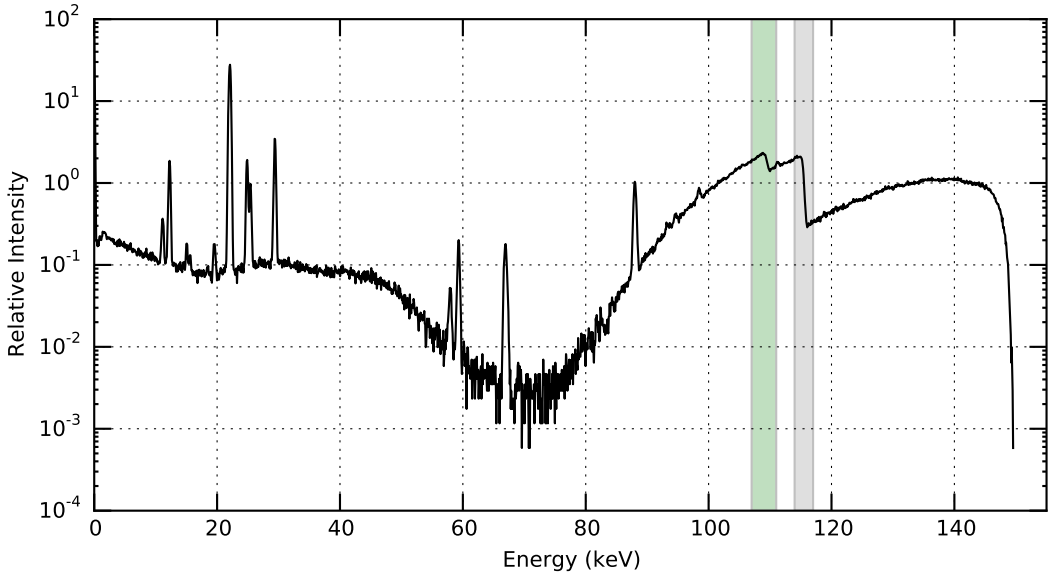


**Figure 7.28:** KED pulse height spectrum of 243.3 g/L U at 4:1 U:Th mass ratio.

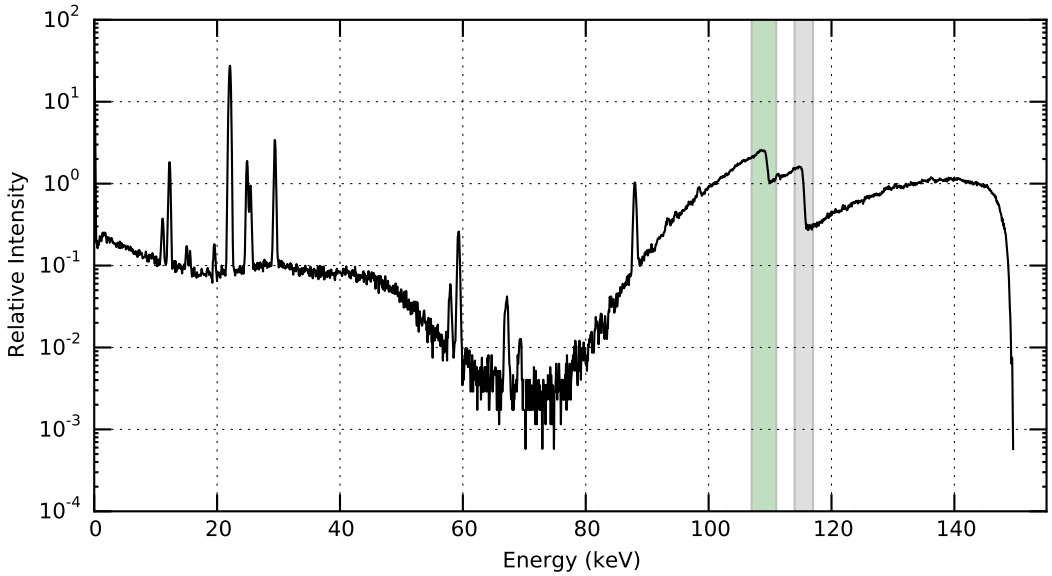


**Figure 7.29:** KED pulse height spectrum of 243.3 g/L U at 2:1 U:Th mass ratio.

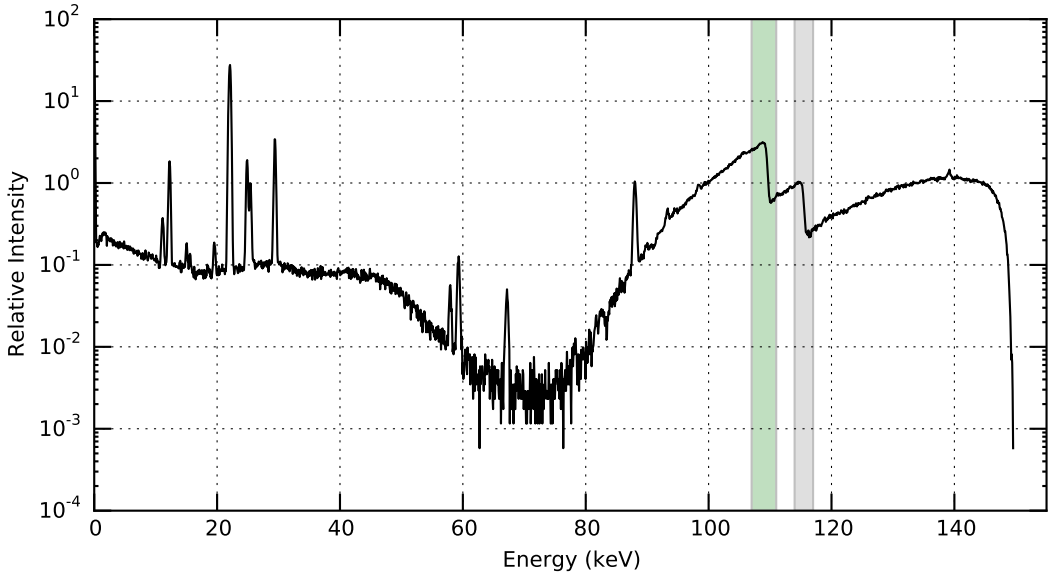


**Figure 7.30:** KED pulse height spectrum of 243.3 g/L U at 1:1 U:Th mass ratio.

The mixed mass ratio K-edge simulations provide an insight as to how more complex mixed actinide samples will behave. Increasing the concentration of thorium closer to the 1:1 mass ratio produces the expected results of forming another K-edge drop. This drop, while not at the same energy as the plutonium K-edge at 121 keV, does provide a similar detector response. The K-edge is not visible until the concentration exceeds 20:1 mass uranium to thorium. At the maximum mass ratio of 1:1 the magnitude of each element's K-edge drop appears to be equal and quite easily distinguished from the other.

The simulations performed on the thorium samples indicate that using this element as a surrogate for plutonium does produce similar effects to the mixed uranium-plutonium samples previously analyzed. The proximity of the peaks in the XRF simulations provide similar interference (i.e. peak overlap) as in the uranium-plutonium cases. Increasing the mass ratio in both the XRF and KED simulations results in similar responses for the thorium as the uranium with regards to K x-ray emission intensities and K-edge continuum drops. The ability to produce these

scoping studies provides the capability to determine what the HKED instrument response would be to samples prepared at a lower cost using thorium surrogates.

## 7.4 Active Sample Source Term

The last study performed was focused on determining the system response to an active sample loaded with fission products. Since pyrochemical reprocessing can handle more recently discharged fuel it would experience a higher gamma-ray signal from these fission products. Development of the gamma-ray source term was accomplished using the ORIGEN tool available as a component of the SCALE code package. A single burnup of 45 $^{GWd}/_{T}$ and a 5 year cooling time were chosen for the study. Gamma rays below 2 MeV were generated and placed into the sample volume in the HKED model. To determine the self-induced x-ray fluorescence source term two simulations, one with a blank sample the other filled with 323.7 $^{g}/_{L}$ of uranium.

To make a direct analysis of the XRF source term a new source term with the gamma-ray source term subtracted was placed in the model and ran to determine what photons originated from x-ray fluorescence. The self induced XRF source term was calculated by subtracting the blank solution from the fission product and uranium loaded solution. The results are shown in Figure 7.31.

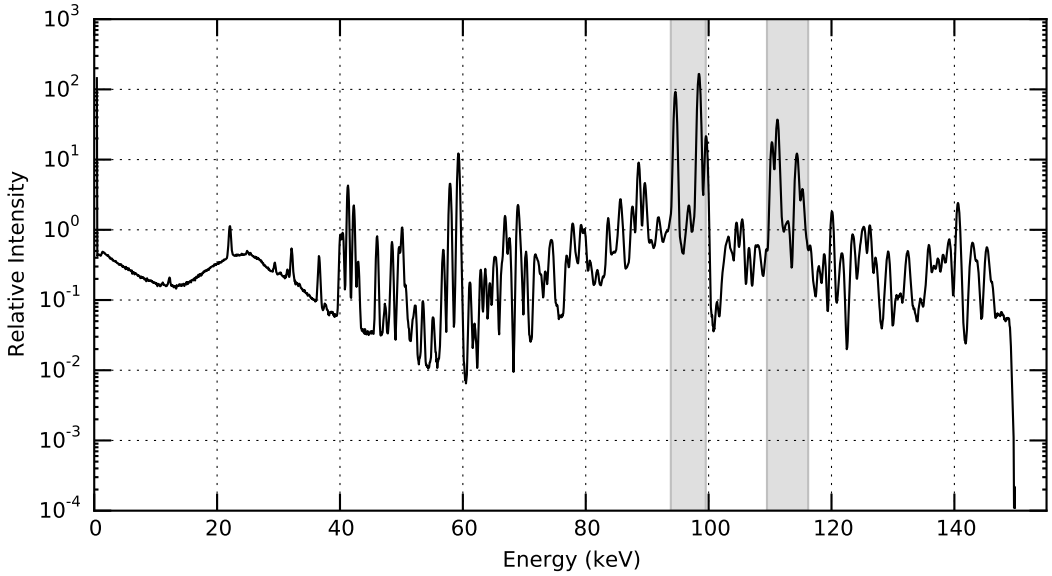


**Figure 7.31:** Fission product source term subtracted XRF pulse height spectrum from 45 $^{GWd}/_{T}$ spent fuel that has cooled for 5 years.

Prominent uranium $K_{\alpha}$ and $K_{\beta}$ emission peaks can be seen in the regions of interest denoted in grey. These simulations were ran without an x-ray source so all x-ray emission peaks are due to excitation from gamma-rays emitted in solution. This source term would interfere with the XRF measurement by increasing the uranium K x-ray peak intensity. An increase in the peak intensity then would lead to a higher than normal reported concentration. Over reporting the concentration of actinide in solution is an obvious safeguards issue and this effect should be further characterized.

# Chapter 8

# Future Work

## 8.1 MCNP Photon Library Inconsistency

The analysis performed in the aqueous validation cases indicate that there is a significant inconsistency in the way MCNP determines x-ray emission and K-edge drop intensity. This is especially evident at the higher concentrations for elements of higher values of Z. While an initial study using three separate elements with different K-edge energies and densities was adequate to identify the issue, a more extensive study is required to fully characterize the issue.

## 8.2 X-ray Tube Characterization

One issue encountered in this work was several inaccuracies in the model of the x-ray tube. While SpekCalc does provide a usable x-ray spectrum, it does not fully match that of the HKED system. The x-ray source in the instrument has never been fully characterized nor modeled. This could be accomplished by first developing a first-principles model of the x-ray tube in MCNP. The resulting spectrum generated by MCNP could then be compared to a data taken with the actual instrument. The most accurate method of measuring the x-ray tube characteristics would be to remove the iron beam filter from the K-edge beamline and operating the x-ray tube at a lower

current. This would allow for characterization of the tube at the operating voltage of 150 kV without damaging the detector.

## 8.3 Gadolinium Filter Modifications

An inconsistency discovered in the MCNP simulations is the behavior of the gadolinium x-ray emission peaks in the XRF spectra. This component of the instrument was not measurable due to the fragility of the filter. A further study of how the system responds to changes in the thickness of the filter is required to determine if the gadolinium filter was not configured correctly from the factory.

## 8.4 Further Burnup and Cooling Times

One advantage of pyroprocessing is that the lack of an organic solvent increases the radiation robustness of the process. As a result, more recently discharged spent fuel can be input into a pyroprocess. The space of varied burnups and cooling times is vast and while an initial study of the HKED instrument's response was completed in support of this work, a more comprehensive study is required. This study should include not only more burnup levels and cooling times but more variety in initial enrichment and fuel types, such as CANDU fuel bundles.

## 8.5 In-Situ Measurements

The ultimate goal for pyroprocessing safeguards is to develop a technique to measure the uranium, plutonium, and other minor actinide content in each process as the process is ongoing. The batch nature of pyroprocessing and the extreme environmental requirements of pyroprocessing result in a great difficulty in making these measurements. Using this model, a method of removing the sample from the electrorefiner should be developed. This could be done by incorporating lessons

learned in using molten salt as a coolant. If the characteristics of molten salt traveling through piping is well understood then a pneumatic system could be developed to draw a sample into a measurement position in a HKED instrument from an electrorefiner.

## 8.6 $^{238}$Pu and Medical Isotope Process Measurements

Medical isotope generation and $^{238}$Pu creation is a core goal for the Radiochemical Engineering Development Center (REDC) at Oak Ridge National Laboratory. Both these efforts involve complex radiochemical processes that must be performed in a hot-cell due to the activity of the components of the process. Developing a method to place the HKED instrument strategically such that process lines could be diverted to the sample position in the instrument. HKED measurements could be used to monitor separations as they proceed and measure product streams.

# Chapter 9

# Conclusions

The model developed in support of this work has proven that HKED can indeed be applied as a safeguards measurement technique in pyrochemical reprocessing. While there are some deficiencies in the model, such as the x-ray intensity issues, it has proven that it is indeed able to predict the HKED system's response to samples from a pyrochemical reprocessing operation.

MCNP simulations of samples taken from both the Mk.4 and Mk.5 electrorefiner were able to predict the system response to a solution of mixed actinides and uranium in a LiCl-KCl eutectic salt. The XRF response clearly shows peaks of not only plutonium and uranium, but also other minor actinides like neptunium and americium. While some emissions are not measurable due to interference from other peaks, for example plutonium $K_{\alpha 2}$ being obscured by uranium $K_{\alpha 1}$. $K_{\alpha}$ peaks of the modeled actinides (uranium, thorium, plutonium, and neptunium) are the most prevalent and thus the most useful for measurements. The large population and tight grouping of the $K_{\beta}$ emissions crowd the spectrum and complicate any kind of measurement for the $K_{\beta}$ peaks except for more simple single or binary systems. The K-edge measurements are still feasible with the slightly denser sample without changing the geometry thus providing the ability to perform sample characterization

without altering the technique already proven for aqueous samples.

One goal of this effort was to extend this measurement technique to new points throughout the pyroprocessing operation. While this would not be an issue in aqueous reprocessing operations due to the similar densities of the samples, pyroprocessing samples come in many different forms and densities. Since a HKED instrument is a substantial investment on the part of a reprocessing facility, it is necessary to find all possible applications for the instrument. The model proves that a simple geometry change enables x-ray fluorescence measurements and, to an extent, K-edge densitometry measurements to be made on samples. While the x-ray fluorescence measurements show great promise, further work should be completed to improve the K-edge measurements. This is a significant advancement since there are currently no non-destructive assay techniques for the voloxidation powder, liquid cathode, or the metallic product. The model indicates that HKED can indeed be applied in other areas thought the process with minimal modification to the instrument.

One major finding in this work was discovery and initial quantification of the MCNP x-ray peak intensity and K-edge drop inconsistency. Both the magnitude of the K x-ray peak emissions and the K-edge drop suffered from an increasing deviation as the concentration of uranium was increased in solution. This deviation was up to 35% higher than the measured data in the highest concentration cases. This is a major issue that must be addressed in before MCNP is to be used for studies involving x-ray fluorescence measurements. While MCNP is currently sufficient to model the HKED system within the scope of this work, it should be further investigated to improve future models of the instrument.

A major challenge in developing effective scoping measurements for the instrument is the selection of actinides to place in the sample. Uranium containing solutions are easier to procure and measure, however the addition of plutonium to the solution begins to increase the overhead cost of sample preparation. One proposed solution to

this was to incorporate thorium as a surrogate for plutonium thus providing the ability to create samples containing a variety of uranium and thorium concentrations without the strenuous materials accountancy requirements inherent in handling plutonium. The model shows the whole the thorium K x-ray peaks and K-edge are quite different in energy than plutonium they provide a useful alternative for further scoping studies. This also opens the opportunity for uranium and thorium containing samples to be created at the Radiochemistry Center for Excellence further lowering the cost of sample preparation.

The final task undertaken was to develop an initial model of how the HKED system would respond to highly active sample and if there would be a noticeable self-induced x-ray fluorescence source term. This study did in fact prove not only that the passive gamma-ray signature response in the XRF detector for a fission product loaded sample is substantial but that there is a significant x-ray source term as a result of the gamma-ray emissions in the sample. A further study of the space of burnups and cooling times, and fission product loading is warranted to determine if the XRF measurement is possible given the self-induced x-ray source term.

# Bibliography

[1] N. US Department of Commerce, "NIST X-Ray Transition Energies," 2014. X-Ray Transition Energies Content Page. xii, 20, 32

[2] R. G. Cochran and N. Tsoulfanidis, *The Nuclear Fuel Cycle: Analsis and Management*. La Grange Park, IL: American Nuclear Society, 2nd ed. xiii, 12, 13

[3] M. F. Simpson and J. D. Law, "Nuclear Fuel Reprocessing," Tech. Rep. INL/EXT-10-17753, Idaho National Laboratory, Idaho Falls, Idaho, Feb. 2010. xiii, 14

[4] T. L. BURR, C. A. COULTER, J. HOWELL, and L. E. WANGEN, "Solution Monitoring: Quantitative and Qualitative Benefits to Nuclear Safeguards," *Journal of Nuclear Science and Technology*, vol. 40, pp. 256–263, Apr. 2003. xiii, 16

[5] V. Bragin, J. Carlson, R. Leslie, R. Schenkel, J. Magill, and K. Mayer, "Proliferation resistance and safeguardability of innovative nuclear fuel cycles," 2001. 1

[6] H. Lee, G.-I. Park, J.-W. Lee, K.-H. Kang, J.-M. Hur, J.-G. Kim, S. Paek, I.-T. Kim, and I.-J. Cho, "Current status of pyroprocessing development at KAERI," *Science and Technology of Nuclear Installations*, vol. 2013, Mar. 2013. 4, 8

[7] P. Dey and N. Bansal, "Spent fuel reprocessing: A vital link in indian nuclear power program," *Nuclear Engineering and Design*, vol. 236, pp. 723–729, Apr. 2006. 4

[8] T. Inoue, T. Koyama, and Y. Arai, "State of the art of pyroprocessing technology in japan," *Energy Procedia*, vol. 7, pp. 405–413, 2011. 4

[9] J. Laidler, J. Battles, W. Miller, J. Ackerman, and E. Carls, "Development of pyroprocessing technology," *Progress in Nuclear Energy*, vol. 31, no. 12, pp. 131–140, 1997. 4

[10] M. Simpson, "Developments of spent nuclear fuel pyroprocessing technology at idaho national laboratory," Tech. Rep. INL/EXT-12-25124, Idaho National Laboratory, Idaho Falls, Idaho, Mar. 2012. 6, 8, 9

[11] R. Benedict, C. Solbrig, B. Westpahal, T. Johnson, S. Li, K. Marsden, and K. Goff, "Pyroprocessing progress at idaho national laboratory," Tech. Rep. INL/COL-07-12983, Idaho National Laboratory, Idaho Falls, Idaho, Sept. 2007. 7

[12] S. X. Li, "Experimental observations on the roles of the cd pool in the mk-IV electrorefiner.pdf," *Nuclear Technology*, vol. 162, May 2008. 7

[13] D. Vaden, S. X. Li, B. R. Westphal, K. B. Davies, T. A. Johnson, and D. M. Pace, "Engineering-scale liquid cadmium cathode experiments," *Nuclear Te*, vol. 162, May 2008. 7

[14] J.-H. Yoo, C.-S. Seo, and E.-H. Kim, "A conceptual study of pyroprocessing for recovering actinides from spent oxide fuels," *Nuclear Engineering and Technology*, vol. 40, pp. 581–592, Dec. 2007. 8

[15] H. Lee, G.-I. Park, K.-H. Kang, J.-M. Hur, J.-G. Kim, D.-H. Ahn, Y.-Z. Cho, and E.-H. Kim, "Pyroprocessing technology development at KAERI," *Nuclear Engineering and Technology*, vol. 43, pp. 317–329, July 2011. 8, 9

[16] E.-H. Kim, G.-I. Park, Y.-Z. Cho, and H.-C. Yang, "A new approach to minimize pyroprocessing waste salts through a series of fission product removal processes," *Nuclear Technology*, vol. 162, pp. 208–219, May 2008. 8, 9

[17] Y.-Z. CHO, G.-H. PARK, H.-C. YANG, D.-S. HAN, H.-S. LEE, and I.-T. KIM, "Minimization of eutectic salt waste from pyroprocessing by oxidative precipitation of lanthanides," *Journal of Nuclear Science and Technology*, vol. 46, no. 10, pp. 1004–1011, 2009. 10

[18] B. Westphal, D. Vaden, S. Li, G. Fredrickson, and R. Mariani, "Fate of noble metals during the pyroprocessing of spent nuclear fuel," Idaho National Laboratory, Sept. 2009. 10

[19] T. Suzuki, M. Tanaka, and S.-i. Koyama, "Recovery of minor actinides from spent molten salt waste and decontamination of molten salt waste," *Progress in Nuclear Energy*, vol. 53, pp. 969–973, Sept. 2011. 10

[20] H.-S. LEE, G.-H. OH, Y.-S. LEE, I.-T. KIM, E.-H. KIM, and J.-H. LEE, "Concentrations of CsCl and SrCl2 from a simulated LiCl salt waste generated by pyroprocessing by using czochralski method," *Journal of Nuclear Science and Technology*, vol. 46, no. 4, pp. 392–397, 2009. 10

[21] S. Priebe and K. Bateman, "The ceramic waste form process at idaho national laboratory," *Nuclear Technology*, vol. 162, pp. 199–207, May 2008. 10

[22] J. R. Allensworth, M. F. Simpson, M.-S. Yim, and S. Phongikaroon, "Investigation of fission product transport into zeolite-a for pyroprocessing waste minimization," *Nuclear Technology*, vol. 181, pp. 337–348, Feb. 2013. 10

[23] E. R. Irish and W. H. Reas, "The Purex Process - A Solvent Extraction Reprocessing Method for Irradiated Uranium," Tech. Rep. HW-49483 A, 1957. 13

[24] W. Beyrich, W. Golly, G. Spannagel, P. D. Bivre, W. H. Wolters, and W. Lycke, "The Assay of Uranium and Plutonium in Reprocessing Input Solutions by Isotope Dilution Mass Spectrometry: Results of the Isotope Dilution Analysis

Measurement Evaluation Programs," *Nuclear Technology*, vol. 75, pp. 73–81, Oct. 1986. 15

[25] C. Moulin, P. Decambox, P. Mauchien, D. Pouyat, and L. Couston, "Direct Uranium(VI) and Nitrate Determinations in Nuclear Reprocessing by Time-Resolved Laser-Induced Fluorescence," *Analytical Chemistry*, vol. 68, pp. 3204–3209, Jan. 1996. 15

[26] K. Song, Y.-I. Lee, and J. Sneddon, "Applications of Laser-Induced Breakdown Spectrometry," *Applied Spectroscopy Reviews*, vol. 32, pp. 183–235, Aug. 1997. 15

[27] H. MINEO, H. ISOGAI, Y. MORITA, and G. UCHIYAMA, "An Investigation into Dissolution Rate of Spent Nuclear Fuel in Aqueous Reprocessing," *Journal of Nuclear Science and Technology*, vol. 41, pp. 126–134, Feb. 2004. 16

[28] T. Burr, A. Coulter, and Wangen, "Simulation and Analysis of Plutonium Reprocessing Plant Data," (Los Alamos, NM), Los Alamos National Laboratory (LANL), July 1996. 16

[29] H. Ottmar and H. Eberle, "The hybrid k-edge/k-XRF densitometer: Principles - design - performance," Tech. Rep. KfK 4590, Kernforschungszentrum Karlsruhe, Feb. 1991. 20

[30] M. Simpson, M. Patterson, J. Lee, Y. Wang, J. Versey, and S. Phongikaroon, "Management of Salt Waste from Electrochemical Processing of Used Nuclear Fuel," in *Proceedings of GLOBAL 2013*, 2013. 20, 71

[31] C. J. Park, J. J. Park, and K. C. Song, "Shielding analysis for an Advanced Voloxidation Process with Monte Carlo calculations," *Annals of Nuclear Energy*, vol. 36, pp. 694–699, 2009. 20

[32] S. Abousahl, B. P. van, H. Eberle, H. Ottmar, B. Lynch, P. Vallet, K. Mayer, and M. Ougier, "Development of quantitative analytical methods for the control

of actinides in a pyrochemical partitioning process," *Radiochimica Acta*, vol. 93, no. 3/2005, pp. 147–153, 2009. 20

[33] A. Berlizov, D. Sharikov, H. Ottmar, H. Eberle, J. Galy, and K. Luetzenkirchen, "A quantitative monte carlo modelling of the uranium and plutonium x-ray fluorescence (XRF) response from a hybrid k-edge/k-XRF densitometer," *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment*, vol. 615, pp. 127–135, Mar. 2010. 23, 32

[34] J. B. Farr, J. Chapman, H. Hall, and R. D. McElroy, "MCNP Model of the Hybrid K-Edge Densitometer System Installed at ORNL," 2011. 23

[35] XCP-3 Monte Carlo Codes, "MCNP6 User's Manual," May 2013. 29

[36] Everett, C. J. and Cashwell, E. D., "MCP code fluorescence-routine revision," Informal Report LA-5240-MS, Los Alamos National Laboratory (LANL), May 1973. 30

[37] H. G. I. Hughes, "Recent developments in low-energy electron/photon transport for MCNP6," Tech. Rep. LA-UR-12-24213, Los Alamos National Laboratory (LANL), Apr. 2013. 31

[38] KAERI, "Table of Nuclides," 2014. 61

# Appendix

# Appendix A

# MCNP Input Decks

## A.1 XRF: Stage 1

### A.1.1 Core Input Deck

```
XRF-S1 MCNP6 HKED Model v.11, MCook
c --- GEOMETRY CARDS ---
c Read in geometry from external file
READ FILE=hked_v11_xrf.geo
c --- END GEOMETRY CARDS ---

c --- DATA CARDS ---
c --- MATERIAL CARDS ---
c U/Pu aqueous solution of 250 g/L U at 100:1 U:Pu ratio
M1     1000.12p  -0.075204988
       7000.12p  -0.032969634
       8000.12p  -0.691028497
      92000.12p  -0.198808793
      94000.12p  -0.001988088
c High density polyethylene
M2    1000.12p   0.667
```

```
      6000.12p   0.333
c Dry air
M3    7000.12p   0.78
      8000.12p   0.21
     18000.12p   0.01
c Aluminum (detector casing)
M4   13000.12p   1
c Tungsten
M5   74000.12p   1
c SST 304L
M6    6000.12p  -0.003
       7000.12p  -0.001
      25000.12p  -0.02
      14000.12p  -0.0075
      15000.12p  -0.00045
      16000.12p  -0.00030
      24000.12p  -0.20
      28000.12p  -0.120
      26000.12p  -0.64775
c Copper (use for x-ray tube until tube is fully modeled)
M8    29000.12p   1
c Beryllium (use for XRF-KED detector windows)
M9     4000.12p   1
c Germanium (use for detector crystals)
M10   32000.12p   1
c Iron for KED filter
M11   26000.12p   1
c Gadolinium
M12   64000.12p   1
c --- END MATERIAL CARDS ---
```

```
c
c --- TALLY CARDS ---
c Read in tally cards from common file
READ FILE=hked_v11_xrf_s1.tal
c --- END TALLY CARDS ---
c
c --- VARIANCE REDUCTION CARDS ---
c Read in variance reduction options from common file
READ FILE=hked_v11_xrf_s1.var
c --- END VARIANCE REDUCTION CARDS ---
c
c --- OPTION CARDS ---
c Read in options from common file
READ FILE=hked_v11_xrf_s1.opt
c --- END OPTION CARDS ---
c
```

### A.1.2 Geometry

```
c HKED Model Geometry, v.11, MCook
c --- CELL CARDS ---
c --- CORE COMPONENTS ---
c Sample solution, force photon collisions in this cell
10    1  -1.1   -100  102                   FCL:P=1     IMP:P=1 IMP:E=1
c Sample vial
11    2   -0.98       100 -101 102                       IMP:P=1 IMP:E=1
c Sample solution, force photon collisions in this cell
c 10    1   -1.1   -100  102                    FCL:P=1  IMP:P=1 IMP:E=1
c Sample vial
c 11    2   -0.98       100 -101 102                       IMP:P=1 IMP:E
    =1
```

```
c Air gap within the sample vial
12    3   -0.001293  -102                                    IMP:P=1 IMP:E
    =1
c X-ray tube
13    8   -8.96      -103  104  105  106                     IMP:P=1 IMP:E
    =1
c X-ray tube interior
14    0              -104  106                               IMP:P=1 IMP:E
    =1
c Beryllium window in x-ray tube
15    9   -1.85      -106                                    IMP:P=1 IMP:E
    =1
c X-ray tube shielding
16    5  -19.25       103 -107 105 -108 109  110  604  -999 IMP:P=1 IMP:E
    =1
c Beam line adjacent to x-ray tube
17    3   -0.001293 (-109:-105)                              IMP:P=1 IMP:E
    =1
c Aluminum beam filter
18    4   -2.70      -110                                    IMP:P=1 IMP:E
    =1
c Air gap filling notch in SS sample tube
19    3   -0.001293  -111                                    IMP:P=1 IMP:E
    =1
c Sample holder tube
20    6   -8.03       101 -112 113  111 -999                IMP:P=1 IMP:E
    =1
c Sample holder tube interior
21    3   -0.001293  -113  101                               IMP:P=1 IMP:E
    =1
```

```
c --- END CORE COMPONENTS ---
c
c --- KED BEAM LINE ---
c Inner KED shield
30    5  -19.25      -300  301  309  304  305  307  308  314  315
                    -999                                  IMP:P=1 IMP:E
                        =1
c Outer KED shield
31    5  -19.25      -302  303  318 -999                   IMP:P=1 IMP:E
    =1
c KED collimator
32    5  -19.25     (-304:-305:-306:-307) 309 310 311 800  IMP:P=1 IMP:E
    =1
c KED collimator beam line tip, sample side
33    3   -0.001293 (-309) #34                             IMP:P=1 IMP:E
    =1
c KED iron beam filter
34   11   -7.874    (-312:-313)                           IMP:P=1 IMP:E
    =1
c KED collimator beam line
35    3   -0.001293  (-310:-311) #34 #91                   IMP:P=1 IMP:E
    =1
c Air gap around collimator
36    3   -0.001293   306 -308 315 #32                     IMP:P=1 IMP:E
    =1
c KED collimator retainer pin recession
37    3   -0.001293  -314  316                            IMP:P=1 IMP:E
    =1
c KED collimator retainer pin, assume 304SS for now
```

```
38    6   -8.03     (-315:-316)                              IMP:P=1 IMP:E
   =1
c Air gap around collimator tip, renumber later
39    3   -0.001293  -317  307                              IMP:P=1 IMP:E
   =1
c Poly detector cup, renumber later
40    2   -0.98       300  303  317 -318 -999               IMP:P=1 IMP:E
   =1
c Beryllium window for KED detector
41    9   -1.85      -319                                   IMP:P=1 IMP:E
   =1
c Aluminum detector casing
42    4   -2.70      -303  319  320  321                    IMP:P=1 IMP:E
   =1
c KED detector interior
43    0             (-320:-321) 319 322                    IMP:P=1 IMP:E
   =1
c KED detector crystal
44   10   -5.323     -322 (320:321) 319                     IMP:P=1 IMP:E
   =1
c --- END KED BEAM LINE ---
c
c --- XRF BEAM LINE ---
c XRF collimator cap (assume tungsten for now)
60    5  -19.25      600 -108 -604 608 609                  IMP:P=1 IMP:E
   =1
c XRF collimator body
61    5  -19.25    (-604:-605:-606:-607) 608 -600 #92       IMP:P=1 IMP:E
   =1
c XRF beam line tip
```

```
62    3   -0.001293 -108  609 -608 #64    #92              IMP:P=1 IMP:E
   =1
c XRF gadolinium filter (confirm thickness & material)
63   12   -7.90     -609                                    IMP:P=1 IMP:E
   =1
c XRF beam line
64    3   -0.001293 -608 -600     #92                      IMP:P=1 IMP:E
   =1
c XRF inner shield
65    5  -19.25     -610  611 605 606 607                   IMP:P=1 IMP:E
   =1
c XRF collimator air gap
66    3   -0.001293 -611  606  605  #61                     IMP:P=1 IMP:E
   =1
c XRF retainer pin recession
c 67
c XRF retainer pin
c 68
c XRF outer shield
69    5  -19.25     -615  616                               IMP:P=1 IMP:E
   =1
c XRF detector cup
70    2   -0.98     -616 617 618 607                        IMP:P=1 IMP:E
   =1
c XRF collimator air gap, detector side
71    3   -0.001293 -618  607          #92                 IMP:P=1 IMP:E
   =1
c XRF beryllium window
72    9   -1.85     -619                                    IMP:P=1 IMP:E
   =1
```

```
c XRF detector casing
73    4   -2.70     -617  619  620  621                      IMP:P=1 IMP:E
    =1
c XRF detector interior
74    0            (-620:-621) 622 619                      IMP:P=1 IMP:E
    =1
c XRF detector crystal
75   10   -5.323    -622 (620:621) 619                       IMP:P=1 IMP:E
    =1
c --- END XRF BEAM LINE ---
c
c --- AUX COMPONENTS ---
c Equipment table
c 90    6   -8.03      -900 -999                            IMP:P=1 IMP:E
    =1
c Microcell at end of KED beam line
91    3   -0.001293   -800       VOL=2.513E-6               IMP:P=1 IMP:E
    =1
c Microcell at end of XRF beam line
92    3   -0.001293   -801       VOL=3.524E-5               IMP:P=1 IMP:E
    =1
c Interior model space, fill with air
98    3   -0.001293   #10 #11 #12 #13 #14 #15 #16 #17 #18 #19
                      #20 #21
                      #30 #31 #32 #33 #34 #35 #36 #37 #38 #39
                      #40 #41 #42 #43 #44
                      #60 #61 #62 #63 #64 #65 #66     #69
                      #70 #71 #72 #73 #74 #75
                      #91 #92 -999                          IMP:P=1 IMP:E
                          =1
```

```
c Exterior of model space
99    0             999                                        IMP:P=0 IMP:E
   =0
c --- END AUX COMPONENTS ---
c --- END CELL CARDS ---

c --- SURFACE CARDS ---
c --- CORE COMPONENTS ---
c Sample vial and solution
100   RCC   6.5     0 -0.98   0      0  3.214  0.7
101   RCC   6.5     0 -1.26   0      0  5.85   0.954
c Air gap inside sample vial above sample, renumber later
102   RCC   6.5     0  2.234  0      0  2.136  0.7
c X-ray tube
103   RCC   0  0 -2   0 0 6      2.5415
c X-ray tube shell
104   RCC   0  0 -1.8    0  0  5.6      2.3415
c Cone cutting off x-ray tube
105   TRC   2.415 0  0    0.13  0  0     0.2  0.21
c Beryllium window
106   RCC   2.335 0  0  0.08  0  0  0.2
c X-ray tube body shielding
107   RCC   0 0 -2   0 0 7.002  7.979
108   PX    3.6955
c Beam line within tungsten x-ray shield
109   RPP   2.5415 3.5815   -0.091 0.091   -0.065  0.065
c Aluminum x-ray filter within tungsten x-ray shield
c 110   RCC   3.5815 0  0      0.114 0  0     1.2835
110   RCC   3.5815 0  0      0.114 0  0     0.80
c Air gap in SS sample tube opposite from aluminum x-ray filter
```

```
111   RPP   3.6955 3.8255   -5.5    1.5     -0.5035 0.5035
c Sample holder tube
112   RPP   3.6955 9.2925 -13.197 15.197 -2     6
113   RPP   3.894  9.094  -11.197 15.197 -1.8015 5.8015
c --- END CORE COMPONENTS ---
c
c --- KED BEAM LINE ---
c Inner KED shield
300   RPP   9.2925 19.4055 -5.005  5.005 -5.005  5.005
301   RPP   9.2925 11.527  -5.005  5.005 -5.005 -3.501
c Outer KED shield
302   RPP  19.4055 29.0145 -5.005  5.005 -5.005 5.005
303   RCC  20.4055 0  0      9.0145 0 0     3.425
c KED collimator
304   RCC   9.2925 0  0      5.883  0  0     0.94
305   RCC  15.1755 0  0      2.6    0  0     1.037
306   RCC  17.7755 0  0      0.552  0  0     0.7295
307   RCC  18.3275 0  0      1.527  0  0     1.073
c Air gap around KED collimator
308   RCC  17.7755 0  0      0.552  0  0     1.037
c KED beam line w/ iron beam filter hole
309   RCC   9.2925 0  0      0.512  0  0     0.2685
310   RCC   9.8045 0  0      1.903  0  0     0.2215
c 311  RCC  11.6955 0  0      8.159  0  0     0.04
311   RCC  11.6955 0  0      8.1585 0  0     0.04
c KED iron beam filter
c old
c 312  RCC   9.3925 0  0      0.402  0  0     0.2585
c 313  RCC   9.7945 0  0      1.803  0  0     0.2115
312   RCC   9.3925 0  0      0.412  0  0     0.2685
```

```
313   RCC   9.7945 0  0      1.803 0  0     0.2215
c KED collimator retainer pin and hole
314   RCC  18.0515 0  4.414  0     0  0.591 0.985
315   RCC  18.0515 0  4.414  0     0 -3.594 0.276
316   RCC  18.0515 0  4.414  0     0  0.575 0.4435
c Air gap around KED collimator tip
317   RCC  19.4055 0  0      1.0   0  0   1.1465
c KED detector cup, polyethylene
318   RCC  19.4055 0  0      9.609 0  0   3.845
c KED detector window, Canberra model GL0510 LE-HPGe
319   RCC  20.4055 0  0      0.015  0  0  2.789
c KED detector housing, assume 2mm casing thickness
320   RCC  20.4205 0  0      0.1985  0  0   2.589
321   RCC  20.619  0  0      8.601   0  0   3.225
c KED detector crystal, Canberra model GL0510 LE-HPGe
322   RCC  20.9055 0  0      1.0     0  0   1.2616
c --- END KED BEAM LINE---
c
c --- XRF BEAM LINE ---
c XRF collimator cap
600      PX   3.4955
601      PX   3.6955
c 602  1   RCC   3.6955 -4.767 0   0.2  0  0   0.9855
c XRF collimator body
604   1   RCC   3.4955 -4.767 0  -11.7835 0  0   0.9855
605   1   RCC  -8.288  -4.767 0   -1.813  0  0   1.4675
606   1   RCC -10.101  -4.767 0   -0.746  0  0   1.053
607   1   RCC -10.847  -4.767 0   -2.487  0  0   1.4675
c XRF beam line
c 608  1   RCC   3.4955 -4.767 0  -16.8295  0  0   0.15
```

```
608   1   RCC   3.4955 -4.767 0  -16.829  0  0   0.15
c XRF gadolinium foil, assume 0.1 mm thickness (Contact Bob & Tyler to
   verify)
609      RCC   3.4955 -2.17   0    0.01    0  0   0.98
c 609      RCC   3.4955 -3.32   0    0.01    0  0   0.98
c XRF inner shield
610   1   RPP -12.496 -8.288 -9.9665 0.4325  -5.005 5.005
c XRF retainer pin air gap
611   1   RCC -10.101  -4.767 0   -0.746  0  0   1.4675
c XRF retainer pin recession
c 612
c XRF collimator retainer pin
c 613
c 614
c XRF outer shield
615   1   RPP  -22.105 -12.496 -9.9665 0.4325  -5.005 5.005
c XRF detector cup
616   1   RCC  -12.496 -4.767 0   -9.609 0  0  3.845
617   1   RCC  -14.001 -4.767 0   -9.038 0  0  3.425
c XRF detector side air gap
618   1   RCC  -12.496 -4.767 0    -1.505 0  0  1.676
c XRF detector window, Canberra model GL0510 LE-HPGe
619   1   RCC  -14.001 -4.767 0     -0.015 0  0  2.789
c XRF detector housing, assume 2mm casing thickness
620   1   RCC  -14.021 -4.767 0       -0.1985  0  0    2.589
621   1   RCC  -14.2145 -4.767 0      -8.601   0  0    3.225
c XRF detector crystal, Canberra model GL0510 LE-HPGe
622   1   RCC  -14.501 -4.767     0   -1.0     0  0    1.2616
c --- END XRF BEAM LINE ---
c
```

```
c Surfaces for DXTRAN spheres (KED & XRF)
c 800      PX  19.854
c 801  1   PX -13.3335
c KED & XRF microcells
800      RCC   19.854  0     0        0.0005 0  0    0.04
801  1   RCC  -13.3335 -4.767 0       -0.0005 0  0    0.15
c --- AUX COMPONENTS ---
c Equipment table
c 900  RPP -20      9.2925 -20     20     -4     -2
c Model exterior boundary
c 999  SO   50
c 999  SX    4  28
999  RCC  4  0   -6   0  0  14    28
c --- END AUX COMPONENTS ---
c --- END SURFACE CARDS ---
```

### A.1.3 Variance Reduction

```
c DXTRAN spheres at both beamline apertures closest to detectors
c KED DXTRAN
c DXT:P 19.8545 0 0   0.04005 0.04005 1E-8 1E-10
c XRF DXTRAN
c DXT:P -9.57 -9.97 0 0.15005 0.15005 1E-7 1E-10
DXT:P -9.57 -9.97 0 0.15005 0.25005 -9.57 -9.97 0 0.35005 0.45005
      -9.57 -9.97 0  0.55005 0.65005 -9.57 -9.97 0 0.75005 0.85005 1E-8 1
         E-10
c Weight window generator and mesh (11 & 18 for KED, 21 & 28 for XRF)
WWG 22 0 0 4J 0
c Energy based weight windows
WWGE:P 0.01 14I 0.150
c Rectangular mesh for KED beamline
```

```
c MESH    GEOM=CYL           ORIGIN=6.5 0 -7    REF=0 0 0
c         AXS=0 0 1          VEC=1 0 0
c         IMESH=1 6 32                                  IINTS= 4 10 2
c         JMESH=6.75 7.25 16                            JINTS=1 10 1
c         KMESH=0.0005 0.00125 0.48 0.52 0.99875 0.9995 1 KINTS=2 2 5 5 5
     2 2
c Cylindrical mesh for XRF beamline
MESH   GEOM=CYL  REF=0 0 0  ORIGIN=3.92 -1.85 -7
       AXS=0 0 1                  VEC=-1 0 0
       IMESH=0.5 1.75 10 32       IINTS=10 10 2 2
       JMESH=6.75 7.25 16         JINTS=1 5 1
       KMESH=0.08472 0.08775 0.25 1 KINTS=1 4 1 3
c Weight window parameters, use values on ESPLT to scale weight windows
c and split particles
c WWP:P 5  3  5  0 -1  0  1  1  0
WWP:P 5  3  5  0 -1  0  0.0003789625 1  0
c Split particles below 40 keV to improve sampling at lower energies
c ESPLT:P   4 0.040  8 0.030
```

### A.1.4 Problem Tallies

```
c --- KED tallies ---
c Stage 1: Particle transport to KED microcell
c FC12 Flux at KED microcell
c F12:P 311.2
c E12  0.00025 559I 0.140
c Stage 2: Detector pulse height tally in KED detector crystal cell
c FC18 Pulse Height Tally in XRF Detector Crystal
c F18:P 44
c E18   1E-10 0.0005 298I  0.15
c --- XRF tallies ---
```

```
c Stage 1: Particle transport to microcell
FC22 Flux at XRF microcell
F22:P 608.2
E22 0 8190I 0.15
SD22 0.141371669
c New F2 Tally to determine total flux
c FC32 Total flux at XRF microcell
c F32:P 608.2
c SD32 0.141371669
c F6 tally tag in sample material
c F16:P  10
c FT16 TAG 3
c FU16 00000.00003 1E10
c Stage 2: Detector pulse height tally in detector crystal cell
c FC28 Pulse Height Tally in XRF Detector Crystal
c F28:P 75
c E28  0 1E-10 0.0005 2047I  0.15
c FT28 GEB 1.9E-4 7E-4 9.5
c --- MESH tallies ---
c Type 1 mesh tally for photons
c TMESH
c  RMESH11:P  FLUX
c  CORA11 -22 100I 22
c  CORB11 -22 100I 22
c  CORC11 -2  10I  6
c ENDMD
c FMESH test
c FMESH14:P GEOM=REC ORIGIN=-22 -22 -2
c      IMESH=32   IINTS=100
c      JMESH=22   JINTS=100
```

```
c     KMESH=6   KINTS=10
c MPLOT FREQ 10000 FMESH 14 BASIS 1 1 0
```

### A.1.5 Options

```
c Physics mode
MODE P E
c Use coherent scattering
PHYS:P J J 0
c Maximum electron energy of 0.25 MeV, single-event history below 2 eV
c this shouldn't trigger single-event history due to CUT card
PHYS:E 0.25 13J 2.0E-6
c Use detailed Landau straggling for single-event history
DBCN 17J 2
c Set specific cutoff energies to bypass defaults, from LANL
   presentation
c Kill photons below 1 keV
CUT:P J 1.0E-3
c Kill electrons below 1 keV
CUT:E J 1.0E-3
c Read in source cards from external file
READ FILE=30GWd_T_3yr.src
c READ FILE=mxr_160_bias_40Kbins.src
c Problem termination condition
NPS 1E3
c Stop run when specific tally error is achieved
c STOP F22 0.0095
c Transform for XRF components
*TR1   0  -1.15 0     31  59  90      121  31  90     90  90  0     -1
c Print detailed output
PRINT -161 -162
```

## A.2 XRF: Stage 2

### A.2.1 Core Input Deck

```
XRF-S2 MCNP6 HKED Model v.11, MCook
c --- GEOMETRY CARDS ---
c Read in geometry from external file
READ FILE=hked_v11_xrf.geo
c --- END GEOMETRY CARDS ---

c --- DATA CARDS ---
c --- MATERIAL CARDS ---
c Aqueous sample solution 300 g/L U
M1    1000.12p  -0.072425000
      7000.12p  -0.031750895
      8000.12p  -0.665484294
     92000.12p  -0.230339811
c High density polyethylene
M2    1000.12p   0.667
      6000.12p   0.333
c Dry air
M3    7000.12p   0.78
      8000.12p   0.21
     18000.12p   0.01
c Aluminum (detector casing)
M4   13000.12p   1
c Tungsten
M5   74000.12p   1
c SST 304L
M6    6000.12p  -0.003
      7000.12p  -0.001
```

```
     25000.12p  -0.02
     14000.12p  -0.0075
     15000.12p  -0.00045
     16000.12p  -0.00030
     24000.12p  -0.20
     28000.12p  -0.120
     26000.12p  -0.64775
c Solid LiCl-KCl (59:41 % mol, Kim et al. 2012)
c M7   3000.12p  -0.055179010
c     19000.12p  -0.501808851
c     17000.12p  -0.003378076
c     92000.12p  -0.435281251
c     94239.12p  -0.004352813
c Copper (use for x-ray tube until tube is fully modeled)
M8   29000.12p   1
c Beryllium (use for XRF-KED detector windows)
M9    4000.12p   1
c Germanium (use for detector crystals)
M10  32000.12p   1
c Iron for KED filter
M11  26000.12p   1
c Gadolinium
M12  64000.12p   1
c Lead
c c M13 82000.12p   1
c --- END MATERIAL CARDS ---
c
c --- TALLY CARDS ---
READ FILE=hked_v11_xrf_s2.tal
c --- END TALLY CARDS ---
```

```
c
c --- VARIANCE REDUCTION CARDS ---
READ FILE=hked_v11_xrf_s2.var
c --- END VARIANCE REDUCTION CARDS ---
c
c --- OPTION CARDS ---
READ FILE=hked_v11_xrf_s2.opt
c Read in source cards from external file
READ FILE=src/hked_v11_xrf_s1_300gL_U.inp.src
c --- END OPTION CARDS ---
```

### A.2.2 Geometry

```
c HKED Model Geometry, v.11, MCook
c --- CELL CARDS ---
c --- CORE COMPONENTS ---
c Sample solution, force photon collisions in this cell
10    1   -1.1     -100  102                          FCL:P=1 IMP:P=1 IMP:E
   =1
c Sample vial
11    2   -0.98       100 -101 102                        IMP:P=1 IMP:E
   =1
c Air gap within the sample vial
12    3   -0.001293  -102                                 IMP:P=1 IMP:E
   =1
c X-ray tube
13    8   -8.96      -103  104  105  106                  IMP:P=1 IMP:E
   =1
c X-ray tube interior
14    0              -104  106                           IMP:P=1 IMP:E
   =1
```

```
c Beryllium window in x-ray tube
15    9   -1.85      -106                                    IMP:P=1 IMP:E
    =1
c X-ray tube shielding
16    5  -19.25       103 -107 105 -108 109  110  604  -999 IMP:P=1 IMP:E
    =1
c Beam line adjacent to x-ray tube
17    3   -0.001293 (-109:-105)                              IMP:P=1 IMP:E
    =1
c Aluminum beam filter
18    4   -2.70      -110                                    IMP:P=1 IMP:E
    =1
c Air gap filling notch in SS sample tube
19    3   -0.001293  -111                                    IMP:P=1 IMP:E
    =1
c Sample holder tube
20    6   -8.03       101 -112 113  111 -999                IMP:P=1 IMP:E
    =1
c Sample holder tube interior
21    3   -0.001293  -113  101                               IMP:P=1 IMP:E
    =1
c --- END CORE COMPONENTS ---
c
c --- KED BEAM LINE ---
c Inner KED shield
30    5  -19.25      -300  301  309  304  305  307  308  314  315
                    -999                                    IMP:P=1 IMP:E
                       =1
c Outer KED shield
```

```
31    5  -19.25      -302  303  318 -999                    IMP:P=1 IMP:E
   =1
c KED collimator
32    5  -19.25     (-304:-305:-306:-307) 309 310 311 800  IMP:P=1 IMP:E
   =1
c KED collimator beam line tip, sample side
33    3   -0.001293 (-309) #34                              IMP:P=1 IMP:E
   =1
c KED iron beam filter
34   11   -7.874    (-312:-313)                            IMP:P=1 IMP:E
   =1
c KED collimator beam line
35    3   -0.001293  (-310:-311) #34 #91                   IMP:P=1 IMP:E
   =1
c Air gap around collimator
36    3   -0.001293   306 -308 315 #32                     IMP:P=1 IMP:E
   =1
c KED collimator retainer pin recession
37    3   -0.001293  -314  316                             IMP:P=1 IMP:E
   =1
c KED collimator retainer pin, assume 304SS for now
38    6   -8.03     (-315:-316)                            IMP:P=1 IMP:E
   =1
c Air gap around collimator tip, renumber later
39    3   -0.001293  -317  307                             IMP:P=1 IMP:E
   =1
c Poly detector cup, renumber later
40    2   -0.98       300  303  317 -318 -999              IMP:P=1 IMP:E
   =1
c Beryllium window for KED detector
```

```
41    9   -1.85      -319                                    IMP:P=1 IMP:E
   =1
c Aluminum detector casing
42    4   -2.70      -303  319  320  321                     IMP:P=1 IMP:E
   =1
c KED detector interior
43    0             (-320:-321) 319 322                     IMP:P=1 IMP:E
   =1
c KED detector crystal
44   10   -5.323     -322 (320:321) 319                      IMP:P=1 IMP:E
   =1
c --- END KED BEAM LINE ---
c
c --- XRF BEAM LINE ---
c XRF collimator cap (assume tungsten for now)
60    5  -19.25      600 -108 -604 608 609                   IMP:P=1 IMP:E
   =1
c XRF collimator body
61    5  -19.25    (-604:-605:-606:-607) 608 -600 #92        IMP:P=1 IMP:E
   =1
c XRF beam line tip
62    3   -0.001293 -108  609 -608 #64    #92               IMP:P=1 IMP:E
   =1
c XRF gadolinium filter (confirm thickness & material)
63   12   -7.90     -609                                    IMP:P=1 IMP:E
   =1
c XRF beam line
64    3   -0.001293 -608 -600     #92                       IMP:P=1 IMP:E
   =1
c XRF inner shield
```

```
65    5  -19.25     -610  611 605 606 607                  IMP:P=1 IMP:E
   =1
c XRF collimator air gap
66    3   -0.001293 -611  606  605  #61                   IMP:P=1 IMP:E
   =1
c XRF retainer pin recession
c 67
c XRF retainer pin
c 68
c XRF outer shield
69    5  -19.25     -615  616                              IMP:P=1 IMP:E
   =1
c XRF detector cup
70    2   -0.98     -616 617 618 607                       IMP:P=1 IMP:E
   =1
c XRF collimator air gap, detector side
71    3   -0.001293 -618  607          #92                IMP:P=1 IMP:E
   =1
c XRF beryllium window
72    9   -1.85     -619                                   IMP:P=1 IMP:E
   =1
c XRF detector casing
73    4   -2.70     -617  619  620  621                    IMP:P=1 IMP:E
   =1
c XRF detector interior
74    0            (-620:-621) 622 619                    IMP:P=1 IMP:E
   =1
c XRF detector crystal
75   10   -5.323    -622 (620:621) 619                     IMP:P=1 IMP:E
   =1
```

```
c --- END XRF BEAM LINE ---
c
c --- AUX COMPONENTS ---
c Equipment table
c 90   6   -8.03      -900 -999                          IMP:P=1 IMP:E
   =1
c Microcell at end of KED beam line
91   3   -0.001293   -800       VOL=2.513E-6             IMP:P=1 IMP:E
   =1
c Microcell at end of XRF beam line
92   3   -0.001293   -801       VOL=3.524E-5             IMP:P=1 IMP:E
   =1
c Interior model space, fill with air
98   3   -0.001293   #10 #11 #12 #13 #14 #15 #16 #17 #18 #19
                     #20 #21
                     #30 #31 #32 #33 #34 #35 #36 #37 #38 #39
                     #40 #41 #42 #43 #44
                     #60 #61 #62 #63 #64 #65 #66     #69
                     #70 #71 #72 #73 #74 #75
                     #91 #92 -999                       IMP:P=1 IMP:E
                        =1
c Exterior of model space
99   0            999                                  IMP:P=0 IMP:E
   =0
c --- END AUX COMPONENTS ---
c --- END CELL CARDS ---

c --- SURFACE CARDS ---
c --- CORE COMPONENTS ---
c Sample vial and solution
```

```
100   RCC   6.5     0 -0.98   0      0  3.214  0.7
101   RCC   6.5     0 -1.26   0      0  5.85   0.954
c Air gap inside sample vial above sample, renumber later
102   RCC   6.5     0  2.234  0      0  2.136  0.7
c X-ray tube
103   RCC   0  0 -2   0 0 6      2.5415
c X-ray tube shell
104   RCC   0  0 -1.8   0  0  5.6      2.3415
c Cone cutting off x-ray tube
105   TRC   2.415 0  0    0.13  0  0     0.2  0.21
c Beryllium window
106   RCC   2.335 0  0  0.08  0  0  0.2
c X-ray tube body shielding
107   RCC   0 0 -2   0 0 7.002  7.979
108   PX    3.6955
c Beam line within tungsten x-ray shield
109   RPP   2.5415 3.5815   -0.091 0.091   -0.065  0.065
c Aluminum x-ray filter within tungsten x-ray shield
c 110   RCC   3.5815 0  0      0.114  0  0     1.2835
110   RCC   3.5815 0  0      0.114  0  0     0.80
c Air gap in SS sample tube opposite from aluminum x-ray filter
111   RPP   3.6955 3.8255   -5.5    1.5     -0.5035 0.5035
c Sample holder tube
112   RPP   3.6955 9.2925 -13.197 15.197 -2     6
113   RPP   3.894  9.094  -11.197 15.197 -1.8015 5.8015
c --- END CORE COMPONENTS ---
c
c --- KED BEAM LINE ---
c Inner KED shield
300   RPP   9.2925 19.4055 -5.005  5.005 -5.005  5.005
```

```
301   RPP   9.2925 11.527  -5.005  5.005 -5.005 -3.501
c Outer KED shield
302   RPP  19.4055 29.0145 -5.005  5.005 -5.005 5.005
303   RCC  20.4055 0  0      9.0145 0 0     3.425
c KED collimator
304   RCC   9.2925 0  0      5.883 0  0     0.94
305   RCC  15.1755 0  0      2.6   0  0     1.037
306   RCC  17.7755 0  0      0.552 0  0     0.7295
307   RCC  18.3275 0  0      1.527 0  0     1.073
c Air gap around KED collimator
308   RCC  17.7755 0  0      0.552 0  0     1.037
c KED beam line w/ iron beam filter hole
309   RCC   9.2925 0  0      0.512 0  0     0.2685
310   RCC   9.8045 0  0      1.903 0  0     0.2215
c 311  RCC  11.6955 0  0      8.159 0  0     0.04
311   RCC  11.6955 0  0      8.1585 0  0     0.04
c KED iron beam filter
c old
c 312  RCC   9.3925 0  0      0.402 0  0     0.2585
c 313  RCC   9.7945 0  0      1.803 0  0     0.2115
312   RCC   9.3925 0  0      0.412 0  0     0.2685
313   RCC   9.7945 0  0      1.803 0  0     0.2215
c KED collimator retainer pin and hole
314   RCC  18.0515 0  4.414  0     0  0.591  0.985
315   RCC  18.0515 0  4.414  0     0 -3.594 0.276
316   RCC  18.0515 0  4.414  0     0  0.575  0.4435
c Air gap around KED collimator tip
317   RCC  19.4055 0  0      1.0   0  0   1.1465
c KED detector cup, polyethylene
318   RCC  19.4055 0  0      9.609 0  0   3.845
```

```
c KED detector window, Canberra model GL0510 LE-HPGe
319   RCC  20.4055 0  0      0.015   0  0   2.789
c KED detector housing, assume 2mm casing thickness
320   RCC  20.4205 0  0      0.1985  0  0   2.589
321   RCC  20.619  0  0      8.601   0  0   3.225
c KED detector crystal, Canberra model GL0510 LE-HPGe
322   RCC  20.9055 0  0      1.0     0  0   1.2616
c --- END KED BEAM LINE---
c
c --- XRF BEAM LINE ---
c XRF collimator cap
600      PX    3.4955
601      PX    3.6955
c 602  1   RCC   3.6955 -4.767 0    0.2  0  0    0.9855
c XRF collimator body
604   1   RCC   3.4955 -4.767 0  -11.7835 0  0   0.9855
605   1   RCC  -8.288  -4.767 0   -1.813  0  0   1.4675
606   1   RCC -10.101  -4.767 0   -0.746  0  0   1.053
607   1   RCC -10.847  -4.767 0   -2.487  0  0   1.4675
c XRF beam line
c 608  1   RCC   3.4955 -4.767 0  -16.8295  0  0   0.15
608   1   RCC   3.4955 -4.767 0  -16.829  0  0   0.15
c XRF gadolinium foil, assume 0.1 mm thickness (Contact Bob & Tyler to
   verify)
609      RCC   3.4955 -2.17  0    0.01    0  0   0.98
c 609      RCC   3.4955 -3.32  0    0.01    0  0   0.98
c XRF inner shield
610   1   RPP -12.496 -8.288 -9.9665 0.4325  -5.005 5.005
c XRF retainer pin air gap
611   1   RCC -10.101  -4.767 0   -0.746  0  0   1.4675
```

```
c XRF retainer pin recession
c 612
c XRF collimator retainer pin
c 613
c 614
c XRF outer shield
615   1   RPP  -22.105 -12.496 -9.9665 0.4325  -5.005 5.005
c XRF detector cup
616   1   RCC  -12.496 -4.767 0   -9.609 0  0  3.845
617   1   RCC  -14.001 -4.767 0   -9.038 0  0  3.425
c XRF detector side air gap
618   1   RCC  -12.496 -4.767 0    -1.505 0  0  1.676
c XRF detector window, Canberra model GL0510 LE-HPGe
619   1   RCC  -14.001 -4.767 0     -0.015 0  0  2.789
c XRF detector housing, assume 2mm casing thickness
620   1   RCC  -14.021 -4.767 0       -0.1985  0  0    2.589
621   1   RCC  -14.2145 -4.767 0      -8.601   0  0    3.225
c XRF detector crystal, Canberra model GL0510 LE-HPGe
622   1   RCC  -14.501 -4.767     0   -1.0     0  0    1.2616
c --- END XRF BEAM LINE ---
c
c Surfaces for DXTRAN spheres (KED & XRF)
c 800      PX  19.854
c 801   1   PX -13.3335
c KED & XRF microcells
800       RCC   19.854  0      0         0.0005 0  0    0.04
801   1   RCC  -13.3335 -4.767 0        -0.0005 0  0    0.15
c --- AUX COMPONENTS ---
c Equipment table
c 900   RPP -20      9.2925 -20      20      -4     -2
```

```
c Model exterior boundary
c 999  SO   50
c 999  SX    4  28
999   RCC  4  0   -6   0  0  14    28
c --- END AUX COMPONENTS ---
c --- END SURFACE CARDS ---
```

### A.2.3 Variance Reduction

```
c DXTRAN spheres at both beamline apertures closest to detectors
c KED DXTRAN
c DXT:P 19.8545 0 0   0.04005 0.04005 1E-8 1E-10
c XRF DXTRAN
c DXT:P -9.57 -9.97 0 0.15005 0.15005 1E-7 1E-10
c Weight window generator and mesh (11 & 18 for KED, 21 & 28 for XRF)
c WWG 22 0 0 4J 0
c Rectangular mesh for KED beamline
c MESH    GEOM=CYL           ORIGIN=6.5 0 -7    REF=0 0 0
c         AXS=0 0 1          VEC=1 0 0
c         IMESH=1 6 32                                  IINTS= 4 10 2
c         JMESH=6.75 7.25 16                            JINTS=1 10 1
c         KMESH=0.0005 0.00125 0.48 0.52 0.99875 0.9995 1 KINTS=2 2 5 5 5
    2 2
c Cylindrical mesh for XRF beamline
c MESH  GEOM=CYL  REF=0 0 0  ORIGIN=3.92 -1.85 -7
c       AXS=0 0 1                   VEC=-1 0 0
c       IMESH=0.5 1.75 10 32        IINTS=10 10 2 2
c       JMESH=6.75 7.25 16          JINTS=1 5 1
c       KMESH=0.08472 0.08775 0.25 1 KINTS=1 4 1 3
c Weight window parameters, use values on ESPLT to scale weight windows
c and split particles
```

```
c WWP:P 5  3  5  0 -1  0  1  1  0
c Split particles below 40 keV to improve sampling at lower energies
c ESPLT:P 2 0.145 4 0.12  8 0.040
c ESPLT:P   4 0.040  8 0.030
```

### A.2.4 Problem Tallies

```
c --- KED tallies ---
c Stage 1: Particle transport to KED microcell
c FC12 Flux at KED microcell
c F12:P 311.2
c E12  0.00025 559I 0.140
c Stage 2: Detector pulse height tally in KED detector crystal cell
c FC18 Pulse Height Tally in XRF Detector Crystal
c F18:P 44
c E18  0 1E-10 0.0005 298I  0.15
c FT18 GEB 1.9E-4 7E-4 9.5
c --- XRF tallies ---
c Stage 1: Particle transport to microcell
c FC22 Flux at XRF microcell
c F22:P 608.2
c E22 0 598I 0.15
c SD22 0.141371669
c Stage 2: Detector pulse height tally in detector crystal cell
FC28 Pulse Height Tally in XRF Detector Crystal
F28:P 75
E28  0 1E-10 0.0005 2047I  0.15
c E28  0 1E-10 0.0005 8188I  0.15
FT28 GEB 1.9E-4 7E-4 9.5
c FT28 PHL 1 26 1 HPG-1
c F6 PHL tally
```

```
c F26:P 75
c --- MESH TALLY CARDS ---
c Type 1 mesh tally for photons
c TMESH
c   RMESH11:P  FLUX
c   CORA11 -22 100I 22
c   CORB11 -22 100I 22
c   CORC11 -2  10I  6
c ENDMD
c FMESH test
c FMESH14:P GEOM=REC ORIGIN=-22 -22 -2
c      IMESH=32   IINTS=100
c      JMESH=22   JINTS=100
c      KMESH=6   KINTS=10
c MPLOT FREQ 10000 FMESH 14 BASIS 1 1 0
```

### A.2.5 Options

```
c Physics mode
MODE P E
c Use coherent scattering
PHYS:P J J 0
c Maximum electron energy of 0.25 MeV, single-event history below 2 eV
c this shouldn't trigger single-event history due to CUT card
PHYS:E 0.25 13J 2.0E-6
c Use detailed Landau straggling for single-event history
DBCN 17J 2
c Set specific cutoff energies to bypass defaults, from LANL
   presentation
c Kill photons below 1 keV
CUT:P J 1.0E-3
```

```
c Kill electrons below 1 keV
CUT:E J 1.0E-3
c Problem termination condition
NPS 5E6
c Stop run when specific tally error is achieved
c STOP F22 0.0095
c Transform for XRF components
*TR1   0  -1.15 0     31  59  90      121  31  90     90  90  0     -1
c Print detailed output
PRINT
```

## A.3 KED: Stage 1

### A.3.1 Core Input Deck

```
KED-S1 MCNP6 HKED Model v.11, MCook
c --- GEOMETRY CARDS ---
c Read in geometry from external file
READ FILE=hked_V11_ked.geo
c --- END GEOMETRY CARDS ---

c --- DATA CARDS ---
c --- MATERIAL CARDS ---
c Aqueous sample solution 300 g/L U
M1    1000.12p  -0.072425000
      7000.12p  -0.031750895
      8000.12p  -0.665484294
     92000.12p  -0.230339811
c High density polyethylene
M2    1000.12p   0.667
      6000.12p   0.333
```

```
c Dry air
M3    7000.12p   0.78
      8000.12p   0.21
     18000.12p   0.01
c Aluminum (detector casing)
M4   13000.12p   1
c Tungsten
M5   74000.12p   1
c SST 304L
M6    6000.12p  -0.003
      7000.12p  -0.001
     25000.12p  -0.02
     14000.12p  -0.0075
     15000.12p  -0.00045
     16000.12p  -0.00030
     24000.12p  -0.20
     28000.12p  -0.120
     26000.12p  -0.64775
c Solid LiCl-KCl (59:41 % mol, Kim et al. 2012)
c M7   3000.12p  -0.055179010
c     19000.12p  -0.501808851
c     17000.12p  -0.003378076
c     92000.12p  -0.435281251
c     94239.12p  -0.004352813
c Copper (use for x-ray tube until tube is fully modeled)
M8   29000.12p   1
c Beryllium (use for XRF-KED detector windows)
M9    4000.12p   1
c Germanium (use for detector crystals)
M10  32000.12p   1
```

```
c Iron for KED filter
M11  26000.12p   1
c Gadolinium
M12  64000.12p   1
c Water
c M13  1000.12p   2
c     8000.12p   1
c --- END MATERIAL CARDS ---
c
c --- TALLY CARDS ---
c --- KED tallies ---
READ FILE=hked_V11_ked_s1.tal
c --- END TALLY CARDS ---
c
c --- VARIANCE REDUCTION CARDS ---
READ FILE=hked_V11_ked_s1.var
c --- END VARIANCE REDUCTION CARDS ---
c
c --- OPTION CARDS ---
READ FILE=hked_V11_ked_s1.opt
c --- END OPTION CARDS ---
```

### A.3.2 Geometry

```
c HKED Model Geometry, v.11, MCook
c --- CELL CARDS ---
c --- CORE COMPONENTS ---
c Sample solution, force photon collisions in this cell
10    1  -1.1    -100  102                                IMP:P=1 IMP:E=1
c Sample vial
11    2   -0.98       100 -101 102                         IMP:P=1 IMP:E=1
```

```
c Sample solution, force photon collisions in this cell
c 10   17  -16.81      -100  102                          IMP:P=1 IMP:E
   =1
c Sample vial
c 11   3   -0.001293     100 -101 102                     IMP:P=1 IMP:E
   =1
c Air gap within the sample vial
12   3   -0.001293  -102                                  IMP:P=1 IMP:E
   =1
c X-ray tube
13   8   -8.96      -103  104  105  106                   IMP:P=1 IMP:E
   =1
c X-ray tube interior
14   0              -104  106                             IMP:P=1 IMP:E
   =1
c Beryllium window in x-ray tube
15   9   -1.85      -106                                  IMP:P=1 IMP:E
   =1
c X-ray tube shielding
16   5  -19.25       103 -107 105 -108 109  110  604  -999 IMP:P=1 IMP:E
   =1
c Beam line adjacent to x-ray tube
17   3   -0.001293 (-109:-105)                           IMP:P=1 IMP:E
   =1
c Aluminum beam filter
18   4   -2.70      -110                                  IMP:P=1 IMP:E
   =1
c Air gap filling notch in SS sample tube
19   3   -0.001293  -111                                  IMP:P=1 IMP:E
   =1
```

```
c Sample holder tube
20    6   -8.03       101 -112 113  111 -999                IMP:P=1 IMP:E
    =1
c Sample holder tube interior
21    3   -0.001293  -113  101                               IMP:P=1 IMP:E
    =1
c --- END CORE COMPONENTS ---
c
c --- KED BEAM LINE ---
c Inner KED shield
30    5  -19.25      -300  301  309  304  305  307  308  314  315
                    -999                                    IMP:P=1 IMP:E
                       =1
c Outer KED shield
31    5  -19.25      -302  303  318 -999                     IMP:P=1 IMP:E
    =1
c KED collimator
32    5  -19.25     (-304:-305:-306:-307) 309 310 311 800   IMP:P=1 IMP:E
    =1
c KED collimator beam line tip, sample side
33    3   -0.001293 (-309) #34                               IMP:P=1 IMP:E
    =1
c KED iron beam filter
34   11   -7.874    (-312:-313)                             IMP:P=1 IMP:E
    =1
c KED collimator beam line
35    3   -0.001293  (-310:-311) #34 #91                     IMP:P=1 IMP:E
    =1
c Air gap around collimator
```

```
36    3   -0.001293   306 -308 315 #32                    IMP:P=1 IMP:E
   =1
c KED collimator retainer pin recession
37    3   -0.001293  -314  316                            IMP:P=1 IMP:E
   =1
c KED collimator retainer pin, assume 304SS for now
38    6   -8.03     (-315:-316)                           IMP:P=1 IMP:E
   =1
c Air gap around collimator tip, renumber later
39    3   -0.001293  -317  307                            IMP:P=1 IMP:E
   =1
c Poly detector cup, renumber later
40    2   -0.98       300  303  317 -318 -999             IMP:P=1 IMP:E
   =1
c Beryllium window for KED detector
41    9   -1.85      -319                                 IMP:P=1 IMP:E
   =1
c Aluminum detector casing
42    4   -2.70      -303  319  320  321                  IMP:P=1 IMP:E
   =1
c KED detector interior
43    0             (-320:-321) 319 322                   IMP:P=1 IMP:E
   =1
c KED detector crystal
44   10   -5.323     -322 (320:321) 319                   IMP:P=1 IMP:E
   =1
c --- END KED BEAM LINE ---
c
c --- XRF BEAM LINE ---
c XRF collimator cap (assume tungsten for now)
```

```
60    5  -19.25      600 -108 -604 608 609                IMP:P=1 IMP:E
   =1
c XRF collimator body
61    5  -19.25    (-604:-605:-606:-607) 608 -600 #92      IMP:P=1 IMP:E
   =1
c XRF beam line tip
62    3   -0.001293 -108  609 -608 #64    #92             IMP:P=1 IMP:E
   =1
c XRF gadolinium filter (confirm thickness & material)
63   12   -7.90     -609                                  IMP:P=1 IMP:E
   =1
c XRF beam line
64    3   -0.001293 -608 -600     #92                     IMP:P=1 IMP:E
   =1
c XRF inner shield
65    5  -19.25     -610  611 605 606 607                 IMP:P=1 IMP:E
   =1
c XRF collimator air gap
66    3   -0.001293 -611  606  605  #61                    IMP:P=1 IMP:E
   =1
c XRF retainer pin recession
c 67
c XRF retainer pin
c 68
c XRF outer shield
69    5  -19.25     -615  616                             IMP:P=1 IMP:E
   =1
c XRF detector cup
70    2   -0.98     -616 617 618 607                      IMP:P=1 IMP:E
   =1
```

```
c XRF collimator air gap, detector side
71    3   -0.001293 -618 607          #92                  IMP:P=1 IMP:E
    =1
c XRF beryllium window
72    9   -1.85     -619                                   IMP:P=1 IMP:E
    =1
c XRF detector casing
73    4   -2.70     -617 619 620 621                       IMP:P=1 IMP:E
    =1
c XRF detector interior
74    0            (-620:-621) 622 619                     IMP:P=1 IMP:E
    =1
c XRF detector crystal
75   10   -5.323    -622 (620:621) 619                     IMP:P=1 IMP:E
    =1
c --- END XRF BEAM LINE ---
c
c --- AUX COMPONENTS ---
c Equipment table
c 90    6   -8.03      -900 -999                          IMP:P=1 IMP:E
    =1
c Microcell at end of KED beam line
91    3   -0.001293   -800       VOL=2.513E-6             IMP:P=1 IMP:E
    =1
c Microcell at end of XRF beam line
92    3   -0.001293   -801       VOL=3.524E-5             IMP:P=1 IMP:E
    =1
c Interior model space, fill with air
98    3   -0.001293   #10 #11 #12 #13 #14 #15 #16 #17 #18 #19
                      #20 #21
```

```
                    #30 #31 #32 #33 #34 #35 #36 #37 #38 #39
                    #40 #41 #42 #43 #44
                    #60 #61 #62 #63 #64 #65 #66      #69
                    #70 #71 #72 #73 #74 #75
                    #91 #92 -999                          IMP:P=1 IMP:E
                       =1
c Exterior of model space
99   0            999                                     IMP:P=0 IMP:E
   =0
c --- END AUX COMPONENTS ---
c --- END CELL CARDS ---

c --- SURFACE CARDS ---
c --- CORE COMPONENTS ---
c Sample vial and solution
100  RCC  6.5    0 -0.98   0     0  3.214 0.7
101  RCC  6.5    0 -1.26   0     0  5.85  0.954
c Air gap inside sample vial above sample, renumber later
102  RCC  6.5    0  2.234  0     0  2.136 0.7
c X-ray tube
103  RCC  0  0 -2   0 0 6      2.5415
c X-ray tube shell
104  RCC  0  0 -1.8   0  0  5.6      2.3415
c Cone cutting off x-ray tube
105  TRC  2.415 0  0    0.13  0  0     0.2  0.21
c Beryllium window
106  RCC  2.335 0  0  0.08  0  0  0.2
c X-ray tube body shielding
107  RCC  0 0 -2  0 0 7.002  7.979
108  PX   3.6955
```

```
c Beam line within tungsten x-ray shield
109   RPP   2.5415 3.5815   -0.091 0.091   -0.065  0.065
c Aluminum x-ray filter within tungsten x-ray shield
c 110  RCC   3.5815 0  0      0.114 0  0     1.2835
110   RCC   3.5815 0  0      0.114 0  0     0.80
c Air gap in SS sample tube opposite from aluminum x-ray filter
111   RPP   3.6955 3.8255   -5.5    1.5    -0.5035 0.5035
c Sample holder tube
112   RPP   3.6955 9.2925 -13.197 15.197 -2     6
113   RPP   3.894  9.094  -11.197 15.197 -1.8015 5.8015
c --- END CORE COMPONENTS ---
c
c --- KED BEAM LINE ---
c Inner KED shield
300   RPP   9.2925 19.4055 -5.005  5.005 -5.005  5.005
301   RPP   9.2925 11.527  -5.005  5.005 -5.005 -3.501
c Outer KED shield
302   RPP  19.4055 29.0145 -5.005  5.005 -5.005 5.005
303   RCC  20.4055 0  0      9.0145 0 0     3.425
c KED collimator
304   RCC   9.2925 0  0      5.883 0  0     0.94
305   RCC  15.1755 0  0      2.6   0  0     1.037
306   RCC  17.7755 0  0      0.552 0  0     0.7295
307   RCC  18.3275 0  0      1.527 0  0     1.073
c Air gap around KED collimator
308   RCC  17.7755 0  0      0.552 0  0     1.037
c KED beam line w/ iron beam filter hole
309   RCC   9.2925 0  0      0.512 0  0     0.2685
310   RCC   9.8045 0  0      1.903 0  0     0.2215
c 311  RCC  11.6955 0  0      8.159 0  0     0.04
```

```
311   RCC  11.6955 0  0      8.1585 0  0     0.04
c KED iron beam filter
c old
c 312  RCC   9.3925 0  0      0.402 0  0     0.2585
c 313  RCC   9.7945 0  0      1.803 0  0     0.2115
312   RCC   9.3925 0  0      0.412 0  0     0.2685
313   RCC   9.7945 0  0      1.803 0  0     0.2215
c KED collimator retainer pin and hole
314   RCC  18.0515 0  4.414  0     0  0.591  0.985
315   RCC  18.0515 0  4.414  0     0 -3.594 0.276
316   RCC  18.0515 0  4.414  0     0  0.575  0.4435
c Air gap around KED collimator tip
317   RCC  19.4055 0  0      1.0   0  0    1.1465
c KED detector cup, polyethylene
318   RCC  19.4055 0  0      9.609 0  0    3.845
c KED detector window, Canberra model GL0510 LE-HPGe
319   RCC  20.4055 0  0      0.015  0  0   2.789
c KED detector housing, assume 2mm casing thickness
320   RCC  20.4205 0  0      0.1985  0  0   2.589
321   RCC  20.619  0  0      8.601   0  0   3.225
c KED detector crystal, Canberra model GL0510 LE-HPGe
322   RCC  20.9055 0  0      1.0     0  0   1.2616
c --- END KED BEAM LINE---
c
c --- XRF BEAM LINE ---
c XRF collimator cap
600      PX   3.4955
601      PX   3.6955
c 602  1   RCC   3.6955 -4.767 0    0.2  0  0    0.9855
c XRF collimator body
```

```
604   1   RCC   3.4955 -4.767 0  -11.7835 0  0   0.9855
605   1   RCC  -8.288  -4.767 0   -1.813  0  0   1.4675
606   1   RCC -10.101  -4.767 0   -0.746  0  0   1.053
607   1   RCC -10.847  -4.767 0   -2.487  0  0   1.4675
c XRF beam line
c 608  1   RCC   3.4955 -4.767 0  -16.8295  0  0   0.15
608   1   RCC   3.4955 -4.767 0  -16.829  0  0   0.15
c XRF gadolinium foil, assume 0.1 mm thickness (Contact Bob & Tyler to
   verify)
609      RCC   3.4955 -2.17  0    0.01    0  0   0.98
c 609      RCC   3.4955 -3.32  0    0.01    0  0   0.98
c XRF inner shield
610   1   RPP -12.496 -8.288 -9.9665 0.4325  -5.005 5.005
c XRF retainer pin air gap
611   1   RCC -10.101  -4.767 0   -0.746  0  0   1.4675
c XRF retainer pin recession
c 612
c XRF collimator retainer pin
c 613
c 614
c XRF outer shield
615   1   RPP  -22.105 -12.496 -9.9665 0.4325  -5.005 5.005
c XRF detector cup
616   1   RCC  -12.496 -4.767 0   -9.609 0  0  3.845
617   1   RCC  -14.001 -4.767 0   -9.038 0  0  3.425
c XRF detector side air gap
618   1   RCC  -12.496 -4.767 0    -1.505 0  0  1.676
c XRF detector window, Canberra model GL0510 LE-HPGe
619   1   RCC  -14.001 -4.767 0     -0.015 0  0  2.789
c XRF detector housing, assume 2mm casing thickness
```

```
620   1   RCC  -14.021 -4.767 0      -0.1985  0  0    2.589
621   1   RCC  -14.2145 -4.767 0     -8.601   0  0    3.225
c XRF detector crystal, Canberra model GL0510 LE-HPGe
622   1   RCC  -14.501 -4.767     0   -1.0     0  0    1.2616
c --- END XRF BEAM LINE ---
c
c Surfaces for DXTRAN spheres (KED & XRF)
c 800      PX  19.854
c 801  1   PX -13.3335
c KED & XRF microcells
800       RCC   19.854  0      0        0.0005 0  0    0.04
801   1   RCC  -13.3335 -4.767 0        -0.0005 0  0    0.15
c --- AUX COMPONENTS ---
c Equipment table
c 900  RPP -20      9.2925 -20     20     -4     -2
c Model exterior boundary
c 999  SO   50
c 999  SX    4  28
999   RCC  4  0   -6   0  0  14    28
c --- END AUX COMPONENTS ---
c --- END SURFACE CARDS ---
```

### A.3.3 Variance Reduction

```
c DXTRAN spheres at both beamline apertures closest to detectors
c KED DXTRAN
DXT:P 19.8545 0  0  0.04005 0.05005 19.8545 0 0  0.06005 0.07005
      19.8545 0  0  8.2445 10.4645 19.8545 0 0  10.5645 10.7945 1E-8 1E
         -10
c XRF DXTRAN
c DXT:P -9.57 -9.97 0 0.15005 0.15005 1E-7 1E-10
```

```
c Weight window generator and mesh (12 & 18 for KED, 22 & 28 for XRF)
WWG 12 0 0 4J 0
c Energy based weight windows
WWGE:P 0.01 14I 0.150
c Rectangular mesh for KED beamline
MESH    GEOM=REC           ORIGIN=-24.5 -29 -6.5   REF=0 0 0
        IMESH=-0.25 9 19.5 33                  IINTS=1 2 5 1
        JMESH=-2.25 -0.09 0.09 2.25 29 JINTS=1 2 1 2 1
        KMESH=-0.06 -0.04 0.04 0.06 9          KINTS=2 1 1 1 2
c Cylindrical mesh for KED beamline
c MESH    GEOM=CYL           ORIGIN=6.5 0 -7    REF=0 0 0
c        AXS=0 0 1          VEC=1 0 0
c        IMESH=1 6 32                                  IINTS= 4 10 2
c        JMESH=6.75 7.25 16                            JINTS=1 10 1
c        KMESH=0.0005 0.00125 0.48 0.52 0.99875 0.9995 1 KINTS=2 2 5 5 5
    2 2
c Cylindrical mesh for XRF beamline
c MESH  GEOM=CYL  REF=0 0 0  ORIGIN=3.92 -1.85 -7
c      AXS=0 0 1                  VEC=-1 0 0
c      IMESH=0.5 1.75 10 32       IINTS=10 10 2 2
c      JMESH=6.75 7.25 16         JINTS=1 5 1
c      KMESH=0.08472 0.08775 0.25 1 KINTS=1 4 1 3
c Weight window parameters, use values on ESPLT to scale weight windows
c and split particles
WWP:P  5  3  5  0 -1  0  1  1  0
c WWP:P 5  3  5  0 -1  0  95633.37987 1  0
c Split particles below 40 keV to improve sampling at lower energies
c ESPLT:P 2 0.145 4 0.12  8 0.040
c ESPLT:P 10 0.12  0.1 0.06
```

### A.3.4 Problem Tallies

```
c Stage 1: Particle transport to KED microcell
FC12 Flux at KED microcell
F12:P 311.2
E12 0 8188I 0.15
SD12 0.502654825E-2
c Stage 2: Detector pulse height tally in KED detector crystal cell
c FC18 Pulse Height Tally in XRF Detector Crystal
c F18:P 44
c E18  0 1E-10 0.00008631 2047I 0.155
c --- XRF tallies ---
c Stage 1: Particle transport to microcell
c FC22 Flux at XRF microcell
c F22:P 608.2
c E12 0 8188I 0.15
c SD22 0.141371669
c Stage 2: Detector pulse height tally in detector crystal cell
c FC28 Pulse Height Tally in XRF Detector Crystal
c F28:P 75
c E28  1E-10 0.0005 298I 0.15
c FT28 GEB 1.9E-4 7E-4 9.5
c --- MESH TALLY CARDS ---
c Type 1 mesh tally for photons
c TMESH
c   RMESH11:P  FLUX
c   CORA11 -22 100I 22
c   CORB11 -22 100I 22
c   CORC11 -2  10I  6
c ENDMD
c FMESH test
```

```
c FMESH14:P GEOM=REC ORIGIN=-22 -22 -2
c     IMESH=32   IINTS=100
c     JMESH=22   JINTS=100
c     KMESH=6   KINTS=10
c MPLOT FREQ 10000 FMESH 14 BASIS 1 1 0
```

### A.3.5 Options

```
c Physics mode
MODE P E
c Use coherent scattering
PHYS:P J J 0
c Maximum electron energy of 0.25 MeV, single-event history below 2 eV
c this shouldn't trigger single-event history due to CUT card
PHYS:E 0.25 13J 2.0E-6
c Use detailed Landau straggling for single-event history
DBCN 17J 2
c Set specific cutoff energies to bypass defaults, from LANL
   presentation
c Kill photons below 1 keV
CUT:P J 1.0E-3
c Kill electrons below 1 keV
CUT:E J 1.0E-3
c Read in source cards from external file
READ FILE=mxr_160_40Kbins.src
c Problem termination condition
NPS 5E8
c Stop run when specific tally error is achieved
c STOP F12 0.0095
c Transform for XRF components
c     Displacement      XX'            YY'          ZZ'    M switch
```

```
*TR1   0  -1.15 0     31  59  90      121  31  90     90  90  0     -1
c Print detailed output
PRINT
```

## A.4 KED: Stage 2

### A.4.1 Core Input Deck

```
KED-S2 MCNP6 HKED Model v.11, MCook
c --- GEOMETRY CARDS ---
c Read in geometry from external file
READ FILE=hked_V11_ked.geo
c --- END GEOMETRY CARDS ---

c --- DATA CARDS ---
c --- MATERIAL CARDS ---
c Aqueous sample solution 300 g/L U
M1    1000.12p  -0.072425000
      7000.12p  -0.031750895
      8000.12p  -0.665484294
     92000.12p  -0.230339811
c High density polyethylene
M2    1000.12p   0.667
      6000.12p   0.333
c Dry air
M3    7000.12p   0.78
      8000.12p   0.21
     18000.12p   0.01
c Aluminum (detector casing)
M4   13000.12p   1
c Tungsten
```

```
M5   74000.12p   1
c SST 304L
M6    6000.12p  -0.003
      7000.12p  -0.001
     25000.12p  -0.02
     14000.12p  -0.0075
     15000.12p  -0.00045
     16000.12p  -0.00030
     24000.12p  -0.20
     28000.12p  -0.120
     26000.12p  -0.64775
c Solid LiCl-KCl (59:41 % mol, Kim et al. 2012)
c M7   3000.12p  -0.055179010
c     19000.12p  -0.501808851
c     17000.12p  -0.003378076
c     92000.12p  -0.435281251
c     94239.12p  -0.004352813
c Copper (use for x-ray tube until tube is fully modeled)
M8   29000.12p   1
c Beryllium (use for XRF-KED detector windows)
M9    4000.12p   1
c Germanium (use for detector crystals)
M10  32000.12p   1
c Iron for KED filter
M11  26000.12p   1
c Gadolinium
M12  64000.12p   1
c Lead
c M13  82000.12p   1
c --- END MATERIAL CARDS ---
```

```
c
c --- TALLY CARDS ---
READ FILE=hked_V11_ked_s2.tal
c --- END TALLY CARDS ---
c
c --- VARIANCE REDUCTION CARDS ---
READ FILE=hked_V11_ked_s2.var
c --- END VARIANCE REDUCTION CARDS ---
c
c --- OPTION CARDS ---
c Read in source cards from external file
READ FILE=src/hked_v11_ked_s1_etest_300gL_U.src
READ FILE=hked_V11_ked_s2.opt
c --- END OPTION CARDS ---
```

### A.4.2 Geometry

```
c HKED Model Geometry, v.11, MCook
c --- CELL CARDS ---
c --- CORE COMPONENTS ---
c Sample solution, force photon collisions in this cell
10    1   -1.1     -100  102                                IMP:P=1 IMP:E
   =1
c Sample vial
11    2   -0.98       100 -101 102                         IMP:P=1 IMP:E
   =1
c Air gap within the sample vial
12    3   -0.001293  -102                                  IMP:P=1 IMP:E
   =1
c X-ray tube
```

```
13    8   -8.96      -103  104  105  106                    IMP:P=1 IMP:E
    =1
c X-ray tube interior
14    0              -104  106                               IMP:P=1 IMP:E
    =1
c Beryllium window in x-ray tube
15    9   -1.85      -106                                    IMP:P=1 IMP:E
    =1
c X-ray tube shielding
16    5  -19.25       103 -107 105 -108 109  110  604  -999 IMP:P=1 IMP:E
    =1
c Beam line adjacent to x-ray tube
17    3   -0.001293 (-109:-105)                             IMP:P=1 IMP:E
    =1
c Aluminum beam filter
18    4   -2.70      -110                                    IMP:P=1 IMP:E
    =1
c Air gap filling notch in SS sample tube
19    3   -0.001293  -111                                   IMP:P=1 IMP:E
    =1
c Sample holder tube
20    6   -8.03       101 -112 113  111 -999                IMP:P=1 IMP:E
    =1
c Sample holder tube interior
21    3   -0.001293  -113  101                               IMP:P=1 IMP:E
    =1
c --- END CORE COMPONENTS ---
c
c --- KED BEAM LINE ---
c Inner KED shield
```

```
30    5  -19.25      -300  301  309  304  305  307  308  314  315
                     -999                                  IMP:P=1 IMP:E
                        =1
c Outer KED shield
31    5  -19.25      -302  303  318 -999                   IMP:P=1 IMP:E
    =1
c KED collimator
32    5  -19.25     (-304:-305:-306:-307) 309 310 311 800  IMP:P=1 IMP:E
    =1
c KED collimator beam line tip, sample side
33    3   -0.001293 (-309) #34                             IMP:P=1 IMP:E
    =1
c KED iron beam filter
34   11   -7.874    (-312:-313)                           IMP:P=1 IMP:E
    =1
c KED collimator beam line
35    3   -0.001293  (-310:-311) #34 #91                   IMP:P=1 IMP:E
    =1
c Air gap around collimator
36    3   -0.001293   306 -308 315 #32                    IMP:P=1 IMP:E
    =1
c KED collimator retainer pin recession
37    3   -0.001293  -314  316                            IMP:P=1 IMP:E
    =1
c KED collimator retainer pin, assume 304SS for now
38    6   -8.03     (-315:-316)                           IMP:P=1 IMP:E
    =1
c Air gap around collimator tip, renumber later
39    3   -0.001293  -317  307                            IMP:P=1 IMP:E
    =1
```

```
c Poly detector cup, renumber later
40    2   -0.98       300  303  317 -318 -999              IMP:P=1 IMP:E
    =1
c Beryllium window for KED detector
41    9   -1.85      -319                                  IMP:P=1 IMP:E
    =1
c Aluminum detector casing
42    4   -2.70      -303  319  320  321                   IMP:P=1 IMP:E
    =1
c KED detector interior
43    0             (-320:-321) 319 322                   IMP:P=1 IMP:E
    =1
c KED detector crystal
44   10   -5.323     -322 (320:321) 319                   IMP:P=1 IMP:E
    =1
c --- END KED BEAM LINE ---
c
c --- XRF BEAM LINE ---
c XRF collimator cap (assume tungsten for now)
60    5  -19.25      600 -108 -604 608 609                IMP:P=1 IMP:E
    =1
c XRF collimator body
61    5  -19.25    (-604:-605:-606:-607) 608 -600 #92      IMP:P=1 IMP:E
    =1
c XRF beam line tip
62    3   -0.001293 -108  609 -608 #64    #92             IMP:P=1 IMP:E
    =1
c XRF gadolinium filter (confirm thickness & material)
63   12   -7.90     -609                                  IMP:P=1 IMP:E
    =1
```

```
c XRF beam line
64    3   -0.001293 -608 -600      #92                    IMP:P=1 IMP:E
    =1
c XRF inner shield
65    5  -19.25     -610  611 605 606 607                  IMP:P=1 IMP:E
    =1
c XRF collimator air gap
66    3   -0.001293 -611  606  605  #61                    IMP:P=1 IMP:E
    =1
c XRF retainer pin recession
c 67
c XRF retainer pin
c 68
c XRF outer shield
69    5  -19.25     -615  616                              IMP:P=1 IMP:E
    =1
c XRF detector cup
70    2   -0.98     -616 617 618 607                       IMP:P=1 IMP:E
    =1
c XRF collimator air gap, detector side
71    3   -0.001293 -618  607          #92                 IMP:P=1 IMP:E
    =1
c XRF beryllium window
72    9   -1.85     -619                                   IMP:P=1 IMP:E
    =1
c XRF detector casing
73    4   -2.70     -617  619  620  621                    IMP:P=1 IMP:E
    =1
c XRF detector interior
```

```
74    0            (-620:-621) 622 619                    IMP:P=1 IMP:E
    =1
c XRF detector crystal
75   10   -5.323    -622 (620:621) 619                    IMP:P=1 IMP:E
    =1
c --- END XRF BEAM LINE ---
c
c --- AUX COMPONENTS ---
c Equipment table
c 90    6   -8.03      -900 -999                          IMP:P=1 IMP:E
    =1
c Microcell at end of KED beam line
91    3   -0.001293   -800       VOL=2.513E-6            IMP:P=1 IMP:E
    =1
c Microcell at end of XRF beam line
92    3   -0.001293   -801       VOL=3.524E-5            IMP:P=1 IMP:E
    =1
c Interior model space, fill with air
98    3   -0.001293   #10 #11 #12 #13 #14 #15 #16 #17 #18 #19
                      #20 #21
                      #30 #31 #32 #33 #34 #35 #36 #37 #38 #39
                      #40 #41 #42 #43 #44
                      #60 #61 #62 #63 #64 #65 #66     #69
                      #70 #71 #72 #73 #74 #75
                      #91 #92 -999                       IMP:P=1 IMP:E
                          =1
c Exterior of model space
99    0             999                                  IMP:P=0 IMP:E
    =0
c --- END AUX COMPONENTS ---
```

```
c --- END CELL CARDS ---


c --- SURFACE CARDS ---
c --- CORE COMPONENTS ---
c Sample vial and solution
100   RCC   6.5     0 -0.98   0      0  3.214  0.7
101   RCC   6.5     0 -1.26   0      0  5.85   0.954
c Air gap inside sample vial above sample, renumber later
102   RCC   6.5     0  2.234  0      0  2.136  0.7
c X-ray tube
103   RCC   0  0 -2   0 0 6      2.5415
c X-ray tube shell
104   RCC   0  0 -1.8    0  0  5.6      2.3415
c Cone cutting off x-ray tube
105   TRC   2.415 0  0    0.13  0  0     0.2  0.21
c Beryllium window
106   RCC   2.335 0  0  0.08  0  0  0.2
c X-ray tube body shielding
107   RCC   0 0 -2   0 0 7.002  7.979
108   PX    3.6955
c Beam line within tungsten x-ray shield
109   RPP   2.5415 3.5815   -0.091 0.091   -0.065  0.065
c Aluminum x-ray filter within tungsten x-ray shield
c 110   RCC   3.5815 0  0       0.114  0  0     1.2835
110   RCC   3.5815 0  0       0.114  0  0     0.80
c Air gap in SS sample tube opposite from aluminum x-ray filter
111   RPP   3.6955 3.8255   -5.5    1.5     -0.5035 0.5035
c Sample holder tube
112   RPP   3.6955 9.2925 -13.197 15.197 -2     6
113   RPP   3.894  9.094  -11.197 15.197 -1.8015 5.8015
```

```
c --- END CORE COMPONENTS ---
c
c --- KED BEAM LINE ---
c Inner KED shield
300   RPP   9.2925 19.4055 -5.005  5.005 -5.005  5.005
301   RPP   9.2925 11.527  -5.005  5.005 -5.005 -3.501
c Outer KED shield
302   RPP  19.4055 29.0145 -5.005  5.005 -5.005 5.005
303   RCC  20.4055 0  0      9.0145 0 0     3.425
c KED collimator
304   RCC   9.2925 0  0      5.883 0  0     0.94
305   RCC  15.1755 0  0      2.6   0  0     1.037
306   RCC  17.7755 0  0      0.552 0  0     0.7295
307   RCC  18.3275 0  0      1.527 0  0     1.073
c Air gap around KED collimator
308   RCC  17.7755 0  0      0.552 0  0     1.037
c KED beam line w/ iron beam filter hole
309   RCC   9.2925 0  0      0.512 0  0     0.2685
310   RCC   9.8045 0  0      1.903 0  0     0.2215
c 311  RCC  11.6955 0  0      8.159 0  0     0.04
311   RCC  11.6955 0  0      8.1585 0  0     0.04
c KED iron beam filter
c old
c 312  RCC   9.3925 0  0      0.402 0  0     0.2585
c 313  RCC   9.7945 0  0      1.803 0  0     0.2115
312   RCC   9.3925 0  0      0.412 0  0     0.2685
313   RCC   9.7945 0  0      1.803 0  0     0.2215
c KED collimator retainer pin and hole
314   RCC  18.0515 0  4.414  0     0  0.591  0.985
315   RCC  18.0515 0  4.414  0     0 -3.594 0.276
```

```
316   RCC  18.0515 0  4.414   0      0  0.575  0.4435
c Air gap around KED collimator tip
317   RCC  19.4055 0  0       1.0    0  0    1.1465
c KED detector cup, polyethylene
318   RCC  19.4055 0  0       9.609  0  0    3.845
c KED detector window, Canberra model GL0510 LE-HPGe
319   RCC  20.4055 0  0       0.015  0  0   2.789
c KED detector housing, assume 2mm casing thickness
320   RCC  20.4205 0  0       0.1985  0  0    2.589
321   RCC  20.619  0  0       8.601   0  0   3.225
c KED detector crystal, Canberra model GL0510 LE-HPGe
322   RCC  20.9055 0  0       1.0     0  0    1.2616
c --- END KED BEAM LINE---
c
c --- XRF BEAM LINE ---
c XRF collimator cap
600       PX    3.4955
601       PX    3.6955
c 602  1   RCC   3.6955 -4.767 0    0.2  0  0    0.9855
c XRF collimator body
604   1   RCC   3.4955 -4.767 0  -11.7835 0  0   0.9855
605   1   RCC  -8.288  -4.767 0   -1.813  0  0   1.4675
606   1   RCC -10.101  -4.767 0   -0.746  0  0   1.053
607   1   RCC -10.847  -4.767 0   -2.487  0  0   1.4675
c XRF beam line
c 608  1   RCC   3.4955 -4.767 0  -16.8295  0  0   0.15
608   1   RCC   3.4955 -4.767 0  -16.829  0  0   0.15
c XRF gadolinium foil, assume 0.1 mm thickness (Contact Bob & Tyler to
   verify)
609       RCC   3.4955 -2.17  0    0.01    0  0   0.98
```

```
c 609      RCC  3.4955 -3.32  0    0.01    0  0   0.98
c XRF inner shield
610   1   RPP -12.496 -8.288 -9.9665 0.4325  -5.005 5.005
c XRF retainer pin air gap
611   1   RCC -10.101  -4.767 0   -0.746  0  0   1.4675
c XRF retainer pin recession
c 612
c XRF collimator retainer pin
c 613
c 614
c XRF outer shield
615   1   RPP  -22.105 -12.496 -9.9665 0.4325  -5.005 5.005
c XRF detector cup
616   1   RCC  -12.496 -4.767 0   -9.609 0  0  3.845
617   1   RCC  -14.001 -4.767 0   -9.038 0  0  3.425
c XRF detector side air gap
618   1   RCC  -12.496 -4.767 0    -1.505 0  0  1.676
c XRF detector window, Canberra model GL0510 LE-HPGe
619   1   RCC  -14.001 -4.767 0     -0.015 0  0  2.789
c XRF detector housing, assume 2mm casing thickness
620   1   RCC  -14.021 -4.767 0       -0.1985  0  0    2.589
621   1   RCC  -14.2145 -4.767 0      -8.601   0  0    3.225
c XRF detector crystal, Canberra model GL0510 LE-HPGe
622   1   RCC  -14.501 -4.767     0   -1.0     0  0    1.2616
c --- END XRF BEAM LINE ---
c
c Surfaces for DXTRAN spheres (KED & XRF)
c 800      PX  19.854
c 801   1   PX -13.3335
c KED & XRF microcells
```

```
800      RCC   19.854  0      0         0.0005 0  0    0.04
801   1  RCC  -13.3335 -4.767 0        -0.0005 0  0    0.15
c --- AUX COMPONENTS ---
c Equipment table
c 900  RPP -20      9.2925 -20     20     -4     -2
c Model exterior boundary
c 999  SO   50
c 999  SX    4  28
999   RCC  4  0   -6   0  0  14    28
c --- END AUX COMPONENTS ---
c --- END SURFACE CARDS ---
```

### A.4.3 Variance Reduction

```
c DXTRAN spheres at both beamline apertures closest to detectors
c KED DXTRAN
c DXT:P 19.8545 0 0   0.04005 0.04005 1E-8 1E-10
c XRF DXTRAN
c DXT:P -9.57 -9.97 0 0.15005 0.15005 1E-7 1E-10
c Weight window generator and mesh (12 & 18 for KED, 22 & 28 for XRF)
c WWG 12 0 0 4J 0
c Rectangular mesh for KED beamline
c MESH    GEOM=CYL           ORIGIN=6.5 0 -7    REF=0 0 0
c         AXS=0 0 1          VEC=1 0 0
c         IMESH=1 6 32                                  IINTS= 4 10 2
c         JMESH=6.75 7.25 16                            JINTS=1 10 1
c         KMESH=0.0005 0.00125 0.48 0.52 0.99875 0.9995 1 KINTS=2 2 5 5 5
     2 2
c Cylindrical mesh for XRF beamline
c MESH  GEOM=CYL  REF=0 0 0  ORIGIN=3.92 -1.85 -7
c       AXS=0 0 1                  VEC=-1 0 0
```

```
c       IMESH=0.5 1.75 10 32        IINTS=10 10 2 2
c       JMESH=6.75 7.25 16          JINTS=1 5 1
c       KMESH=0.08472 0.08775 0.25 1 KINTS=1 4 1 3
c Weight window parameters, use values on ESPLT to scale weight windows
c and split particles
c WWP:P 5  3  5  0 -1  0  1  1  0
c Split particles below 40 keV to improve sampling at lower energies
c ESPLT:P 2 0.145 4 0.12  8 0.040
c ESPLT:P   4 0.040  8 0.030
```

### A.4.4 Problem Tallies

```
c --- KED tallies ---
c Stage 1: Particle transport to KED microcell
c FC12 Flux at KED microcell
c F12:P 311.2
c E12 0 598I 0.15
c SD12 0.502654825E-2
c Stage 2: Detector pulse height tally in KED detector crystal cell
FC18 Pulse Height Tally in XRF Detector Crystal
F18:P 44
E18   0 1E-10 0.00008631 2047I 0.155
FT18 GEB 1.9E-4 7E-4 9.5
c --- XRF tallies ---
c Stage 1: Particle transport to microcell
c FC22 Flux at XRF microcell
c F22:P 608.2
c E12 0 5998I 0.15
c SD22 0.141371669
c Stage 2: Detector pulse height tally in detector crystal cell
c FC28 Pulse Height Tally in XRF Detector Crystal
```

```
c F28:P 75
c E28  1E-10 0.0005 298I  0.15
c FT28 GEB 1.9E-4 7E-4 9.5
c --- MESH tally cards ---
c Type 1 mesh tally for photons
c TMESH
c   RMESH11:P  FLUX
c   CORA11 -22 100I 22
c   CORB11 -22 100I 22
c   CORC11 -2  10I  6
c ENDMD
c FMESH test
c FMESH14:P GEOM=REC ORIGIN=-22 -22 -2
c      IMESH=32   IINTS=100
c      JMESH=22   JINTS=100
c      KMESH=6   KINTS=10
c MPLOT FREQ 10000 FMESH 14 BASIS 1 1 0
```

### A.4.5 Options

```
c Physics mode
MODE P E
c Use coherent scattering
PHYS:P J J 0
c Maximum electron energy of 0.25 MeV, single-event history below 2 eV
c this shouldn't trigger single-event history due to CUT card
PHYS:E 0.25 13J 2.0E-6
c Use detailed Landau straggling for single-event history
DBCN 17J 2
c Set specific cutoff energies to bypass defaults, from LANL
   presentation
```

```
c Kill photons below 1 keV
CUT:P J 1.0E-3
c Kill electrons below 1 keV
CUT:E J 1.0E-3
c Problem termination condition
NPS 2E6
c Stop run when specific tally error is achieved
c STOP F22 0.0095
c Transform for XRF components
*TR1   0  -1.15 0     31  59  90      121  31  90     90  90  0     -1
c Print detailed output
PRINT
```

# Appendix B

# Sample Compositions

## B.1 Aqueous Uranium Solutions

```
c 001gL_U
c Exact sample solution:    1.072 g/L U
M1    1000.12p  -0.094737278
      7000.12p  -0.039514509
      8000.12p  -0.864740078
     92000.12p  -0.001008135

c 005gL_U
c Exact sample solution:    5.459 g/L U
M1    1000.12p  -0.094347921
      7000.12p  -0.03935211
      8000.12p  -0.861186117
     92000.12p  -0.005113852

c 015gL_U
c Exact sample solution:    15.895 g/L U
```

```
M1    1000.12p  -0.093433735
      7000.12p  -0.038970807
      8000.12p  -0.85284164
     92000.12p  -0.014753818

c 045gL_U
c Exact sample solution:    48.12 g/L U
M1    1000.12p  -0.090713513
      7000.12p  -0.037836215
      8000.12p  -0.82801208
     92000.12p  -0.043438193

c 100gL_U
c Exact sample solution:    107.1 g/L U
M1    1000.12p  -0.086103383
      7000.12p  -0.035913349
      8000.12p  -0.785931872
     92000.12p  -0.092051397

c 150gL_U
c Exact sample solution:    160.8 g/L U
M1    1000.12p  -0.082273825
      7000.12p  -0.034316057
      8000.12p  -0.750976549
     92000.12p  -0.132433569

c 200gL_U
c Exact sample solution:    214.5 g/L U
M1    1000.12p  -0.07875134
```

```
      7000.12p  -0.032846844
      8000.12p  -0.718824116
     92000.12p  -0.1695777

c 250gL_U
c Exact sample solution:    268.4 g/L U
M1    1000.12p  -0.075488778
      7000.12p  -0.031486043
      8000.12p  -0.689044202
     92000.12p  -0.203980976

c 300gL_U
c Exact sample solution:    323.7 g/L U
M1    1000.12p  -0.072393804
      7000.12p  -0.030195143
      8000.12p  -0.660793990
     92000.12p  -0.236617063
```

## B.2 Aqueous Uranium and Plutonium Solutions

```
c 100gL_U-Pu
c Exact sample solution: 107.5 g/L U at 103:1 U:Pu ratio
M1    1000.12p  -0.085996081
      7000.12p  -0.035868594
      8000.12p  -0.784952449
     92000.12p  -0.092286886
     94000.12p  -0.000895989

c 150gL_U-Pu
```

```
c Exact sample solution: 160.8 g/L U at 103:1 U:Pu ratio
M1    1000.12p  -0.082167304
      7000.12p  -0.034271627
      8000.12p  -0.750004246
     92000.12p  -0.132272622
     94000.12p  -0.0012842

c 200gL_U-Pu
c Exact sample solution: 213.0 g/L U at 102.4:1 U:Pu ratio
M1    1000.12p  -0.078714883
      7000.12p  -0.032831638
      8000.12p  -0.718491344
     92000.12p  -0.168318401
     94000.12p  -0.001643734

c 250gL_U-Pu
c Exact sample solution: 243.3 g/L U at 82.81:1 U:Pu ratio
M1    1000.12p  -0.076799268
      7000.12p  -0.032032643
      8000.12p  -0.701006055
     92000.12p  -0.187893068
     94000.12p  -0.002268966
```

## B.3 Salt Solutions

```
c Actinide/KCl-LiCl cocktail Mk.5 ER (Simpson et. al., GLOBAL 2013)
M1    3000.12p   -0.06309180
     19000.12p   -0.24178038
     17000.12p   -0.60773692
```

```
     11000.12p   -0.02277697
     37000.12p   -0.00004170
     38000.12p   -0.00010778
     39000.12p   -0.00007376
     55000.12p   -0.00054549
     56000.12p   -0.00034624
     57000.12p   -0.00020219
     58000.12p   -0.00036782
     59000.12p   -0.00016754
     60000.12p   -0.00061590
     61000.12p   -0.00001263
     62000.12p   -0.00016810
     63000.12p   -0.00000877
     64000.12p   -0.00000990
     93000.12p   -0.00005025
     92000.12p   -0.02184114
     94000.12p   -0.04004567
     95000.12p   -0.00000904

c Actinide/KCl-LiCl cocktail Mk.4 ER (Simpson et. al., GLOBAL 2013)
M1    3000.12p   -0.05213168
     19000.12p    -0.20349412
     17000.12p    -0.57866553
     11000.12p   -0.03863036
     37000.12p   -0.00118040
     38000.12p   -0.00275263
     39000.12p   -0.00159365
     55000.12p   -0.00978889
     56000.12p   -0.00474844
```

```
     57000.12p   -0.00482539
     58000.12p   -0.00920971
     59000.12p   -0.00457035
     60000.12p   -0.01571402
     61000.12p   -0.00032998
     62000.12p   -0.00304574
     63000.12p   -0.00014237
     64000.12p   -0.00009664
     93000.12p   -0.00097326
     92000.12p   -0.04734550
     94000.12p   -0.02075410
     95000.12p   -0.00000723
```

## B.4 Surrogate Salt Solutions

```
c 100gL_U-Th
c Exact sample solution: 107.5 g/L U at 103:1 U:Pu ratio
M1    1000.12p  -0.085996081
      7000.12p  -0.035868594
      8000.12p  -0.784952449
     92000.12p  -0.092286886
     90000.12p  -0.000895989

c 150gL_U-Th
c Exact sample solution: 160.8 g/L U at 103:1 U:Pu ratio
M1    1000.12p  -0.082167304
      7000.12p  -0.034271627
      8000.12p  -0.750004246
     92000.12p  -0.132272622
```

```
     90000.12p  -0.0012842

c 200gL_U-Th
c Exact sample solution: 213.0 g/L U at 102.4:1 U:Pu ratio
M1    1000.12p  -0.078714883
      7000.12p  -0.032831638
      8000.12p  -0.718491344
     92000.12p  -0.168318401
     90000.12p  -0.001643734

c 250gL_U-Th
c Exact sample solution: 243.3 g/L U at 82.81:1 U:Pu ratio
M1    1000.12p  -0.076799268
      7000.12p  -0.032032643
      8000.12p  -0.701006055
     92000.12p  -0.187893068
     90000.12p  -0.002268966
```

## B.5 Voloxidation Powder

```
c Powder sample from voloxidation stage, rho = 3.29 g/cc (Park et al., 2009)
M1    92238.12p   -0.8483859
      92235.12p   -0.0098244
      94239.12p   -0.0057936
      92236.12p   -0.0055182
      54136.12p   -0.0026994
      94240.12p   -0.0023689
      54134.12p   -0.0018914
      60144.12p   -0.0015333
```

58140.12p -0.0016252

56138.12p -0.0016160

57139.12p -0.0015242

58142.12p -0.0014232

59141.12p -0.0014048

54132.12p -0.0013956

55133.12p -0.0013956

42100.12p -0.0011661

95241.12p -0.0001359

42098.12p -0.0010375

40096.12p -0.0010100

60143.12p -0.0010008

40094.12p -0.0009733

37102.12p -0.0009733

43099.12p -0.0009641

42097.12p -0.0009549

37101.12p -0.0009549

42095.12p -0.0009457

60146.12p -0.0008860

56137.12p -0.0001093

60145.12p -0.0008319

40092.12p -0.0008144

40091.12p -0.0007630

93237.12p -0.0006354

44104.12p -0.0006574

94242.12p -0.0006271

55135.12p -0.0006042

40093.12p -0.0005922

39089.12p -0.0005904

```
      45103.12p   -0.0005518
      54131.12p   -0.0004894
      55137.12p   -0.0014966
       7000.12p   -0.0000092
       8000.12p   -0.0918167
      18000.12p   -0.0000092
```

## B.6 Liquid Cathode

```
c Liquid cadmium cathode 4.68 wt.% Pu, rho = 8.549 g/cc (Iizuka et al.,2001)
M1     48000.12p   -0.9532
       94000.12p   -0.0468
```

# Appendix C

# Analysis Tools

## C.1 Hybrid K-edge Python Analysis Tools

### C.1.1 Aqueous Samples

```
# -*- coding: utf-8 -*-
# HPAT: HKED Python Analysis Tool
# Version 2.0
#   Matthew T. Cook
# 12 March 2014
# Department of Nuclear Engineering
# University of Tennessee, Knoxville

# import modules
import os as os
import numpy as np
import matplotlib.pyplot as plt
import warnings
#import math as math
#from matplotlib.backends.backend_pdf import PdfPages as pp
```

```
#from matplotlib.ticker import MultipleLocator

# Warning control
# Set script to ignore runtime warnings from modelResid divide by zero
warnings.simplefilter("ignore")

# Clear the screen
os.system('clear')

# Identify script
print " ---------------------------------"
print "| HKED Python Analysis Tool v. 2.0|"
print " ---------------------------------"

# Run in automatic
auto = "n" # "y" or "n"

# Save plot files?
saveFile = "y" # "y" or "n"

plotResults = "y"

# Auto analysis
if auto == "y":
    # Define the U concentrations
    conc = ("001","005","015","045","100","150","200","250","300")
    # Define the Pu concentrations
    #conc = ("100","150","250")
    # Run type
    run  = "xrf"  # "xrf" or "ked"
```

```
    # Stage type
    stage = "s2"  # "s1" or "s2"
    # Sample type
    sType = "U"   # Options: "U" or "U-Pu"

elif auto == "n":
    conc = "250"
    run  = "xrf"  # "xrf" or "ked"
    #sampType = "u" # "u" or "upu"
    stage = "s2"  # "s1" or "s2"
    # Hardcode sample type
    sType = "U-Pu"  # Options: "U" or "U-Pu"
    cat = "aqueous"

if stage == "s1":
    outdir = "s1_cases"
elif stage == "s2":
    outdir = "s2_cases"

# Hardcode file name in for now
specFile = "data/"+run+"_"+conc+"gL_"+sType+".txt"
#specFile = "data/old/300gL_ked_u.txt"

fileName = outdir+"/"+run+"/"+cat+"/hked_v11_"+run+"_"+stage+"_"+conc+"
    gL_"+sType+".inp.o"

#fileName = "s1_cases/xrf/surrogates/hked_v11_xrf_s1_250gL_U-Th.inp.o"


name = conc+"gL_"+run+"_"+sType
```

```
# Hardcode figure file names
figName = name + "_full.pdf"
figNameROI = name + "_detail.pdf"
figName2 = name + "_full.png"
figNameROI2 = name + "_detail.png"

# Open and create files needed by this script
# Open the input file for reading
f = open(fileName, "r")

# rename the input file and open it as a post processed output
base = os.path.splitext(fileName)[0]
outFile = base + ".opp"
sourceFile = base + ".src"
if stage == "s2":
    resultsFile = base + ".res"
    r = open(resultsFile, "w")
    r.write(name+",")

# create and open the output files for writing
o = open(outFile, "w")
#s = open(sourceFile, "wb")

# Determine the run type and search terms

# Tell the user what's going on
print "\nProcessing MCNP output:", fileName
# Search the file for XRF-RUN or KED-RUN tags
with f as search:
```

```
    for line in search:
        # Remove '\n' at end of line
        line = line.rstrip()
        if "xrf-s1" in line:
            aType = "XRF-S1"
            print "Run Type:", aType
            break
        elif "xrf-s2" in line:
            aType = "XRF-S2"
            print "Run Type:", aType
            break
        elif "ked-s1" in line:
            aType = "KED-S1"
            print "Run Type:", aType
            break
        elif "ked-s2" in line:
            aType = "KED-S2"
            print "Run Type:", aType
            break

if aType == "XRF-S2" or aType == "KED-S2":
    # Ask the user what to do
    #plotResults = "n" #raw_input("Plot the results? (y/n): ")
    #plotResults = "n"

    # Ask user if a measured spectrum is to be analyzed
    importSpectrum = "y" #raw_input("Analyze a measured spectrum? (y/n):
        ")
if aType == "XRF-S1" or aType == "KED-S1":
    importSpectrum = "n"
```

```
    plotResults = "n"


# Tell the user where the extracted spectrum is
print "MCNP tally extracted to:",outFile

# Set the tally surface/volume search string
if aType == "XRF-S1":
    searchPhrase = " surface 608.2"
    print "Searched MCNP output for:", searchPhrase
elif aType == "XRF-S2":
    searchPhrase = " cell 75"
    print "Searched MCNP output for:", searchPhrase
elif aType == "KED-S1":
    searchPhrase = " surface 311.2"
    print "Searched MCNP output for:", searchPhrase
elif aType == "KED-S2":
    searchPhrase = " cell 44"
    print "Searched MCNP output for:", searchPhrase

# Create an output file and extract spectrum from MCNP output

if aType == "XRF-S1" or aType == "KED-S1":
    # Identify Stage 1 analysis
    print "Extracting Stage 1 F2 tally..."
    # Search for the specified string in the input file
    with open(fileName) as search:
        for line in search:
            # Remove '\n' at end of line
            line = line.rstrip()
```

```
            if searchPhrase in line:
                #for x in range (0,2053):
                #for x in range (0,602):
                #for x in range (0,2051):
                for x in range (0,8194):
                        line = search.next()
                        # Remove the spaces between tally values
                        # and replace with commas
                        line = line.replace(' ','')
                        line = line.replace(' ',',')
                        line = line.replace(' ',',')
                        line = line.replace(',,energy,','energy')
                        line = line.replace(',,total,,',',total,')
                        # Write the data to analysis file in CSV format
                        o.write(line)
    # Close and reopen output file to flush the buffer
    o.close()
    o = open(outFile,"r")

# Create source file from Stage 1 tally for Stage 2 source

# write the Stage 2 SDEF options to the new source file with a Cd-109
    check source
if aType == "XRF-S1":
    s = open(sourceFile, "wb")
    s2sdef = "SDEF VEC=-0.692 -0.721 0 DIR=1 POS=D1 ERG=FPOS=D2 PAR=2
        ARA=0.001 \n"
    s2sdef = s2sdef + "SI1 L -9.57 -9.97 0  -9.57 -9.97 0 \n"
    s2sdef = s2sdef + "SP1  1              1E-3 \n"
    s2sdef = s2sdef + "DS2 S 4             3 \n"
```

```
    s2sdef = s2sdef + "SI3 L 0.02199 0.022163 0.024912 0.02943 0.025455
        0.0880336 \n"
    s2sdef = s2sdef + "SP3 D 0.298  0.561   0.048   0.092  0.0231
        0.0370"
elif aType =="KED-S1":
    s = open(sourceFile, "wb")
    s2sdef = "SDEF VEC=1 0 0 DIR=1 POS=D1 ERG=FPOS=D2 PAR=2 ARA=0.001 \n
        "
    s2sdef = s2sdef + "SI1 L 19.8545 0 0   19.8545 1 0 \n"
    s2sdef = s2sdef + "SP1   1               2E-1 \n"
    s2sdef = s2sdef + "DS2 S 4               3 \n"
    s2sdef = s2sdef + "SI3 L 0.02199 0.022163 0.024912 0.02943 0.025455
        0.0880336 \n"
    s2sdef = s2sdef + "SP3 D 0.298  0.561   0.048   0.092  0.0231
        0.0370"

if aType == "XRF-S1" or aType == "KED-S1":
    print "Writing Stage 2 source file..."
    # write source file header
    sheader = "c Source file processed from output: "
    s.write(sheader + fileName + "\n")
    s.write(s2sdef + "\n")

    # import the source data from the preprocessed output file
    sourcein = np.genfromtxt(outFile, delimiter=",", skip_header=1,
        skip_footer=1,
            usecols=(0,1))
    mcnpEnergy = np.genfromtxt(outFile, delimiter=",", skip_header=1,
        skip_footer=1,
            usecols=(0))
```

```
mcnpTally = np.genfromtxt(outFile, delimiter=",", skip_header=1,
    skip_footer=1,
        usecols=(1))
mcnpError= np.genfromtxt(outFile, delimiter=",", skip_header=1,
    skip_footer=1,
        usecols=(2))

# sum the source and copy the energies
sumsource = sum(sourcein[:,1])
sourceenergy = sourcein[:,0]

# normalize the Stage 1 tally
for x in range(0,len(sourcein)):
    sourcenorm = sourcein[:,1]/sumsource

# Polynomial fit parameters
#p1  =   5.347e-06
#p2  =   5.042e-06
#p3  =  -3.899e-05
#p4  =  -3.092e-05
#p5  =   8.373e-05
#p6  =   5.541e-05
#p7  =  -5.538e-05
#p8  =   -5.86e-05
#p9  =  -0.0002095
#p10 = -0.0002799
p1 = -0.2099
p2 =  0.07759
p3 = -0.0003616
p4 = -2.786E-5
```

```
# Correct the energy
for i in range(0,len(sourceenergy)):
    sourceenergy[i] = sourceenergy[i] - (p1*sourceenergy[i]**3 + p2*
        sourceenergy[i]**2 + p3*sourceenergy[i] + p4)

# write source energies
for x in range(0,len(sourcein)):
    if x == 0:
        #soutline1 = "SI1 L "
        # Use SI1 L for source with Cd-109
        soutline1 = "SI4 L "
        soutline2 = str(sourceenergy[x])
        s.write(soutline1)
        s.write(soutline2 + "\n")
    else:
        soutline = ("     " + str(sourceenergy[x]))
        s.write(soutline + "\n")

# write the source intensities
for x in range(0,len(sourcein)):
    if x == 0:
        #soutline1 = "SP1 D "
        # Use SP4 D for source with Cd-109
        soutline1 = "SP4 D "
        soutline2 = str(sourcenorm[x])
        s.write(soutline1)
        s.write(soutline2 + "\n")
    else:
        soutline = ("     " + str(sourcenorm[x]))
```

```
            s.write(soutline + "\n")

    # tell user where the source file was written
    print "Stage 1 tally written to Stage 2 source file:", sourceFile
    s.close()

    errorin = np.genfromtxt(outFile, delimiter=",", skip_header=1,
        skip_footer=1,
            usecols=(0,2))

# Import data from Stage 2 runs

if aType == "XRF-S2" or aType == "KED-S2":
    print "Extracting Stage 2 F8 tally..."
    with open(fileName) as search:
        for line in search:
            line = line.rstrip() # remove '\n' at end of line
            if searchPhrase in line:
                for x in range (0,2052):
                    line = search.next()
                    # remove the spaces between tally values
                    # and replace with commas
                    line = line.replace('  ','')
                    line = line.replace(' ',',')
                    line = line.replace(' ',',')
                    line = line.replace(',,energy,','energy')
                    line = line.replace(',,total,,',',total,')
                    # write the data to analysis file in CSV format
                    o.write(line)
```

```
# Close and reopen the file to flush the buffer
    o.close()
    o = open(outFile,"r")
# Normalize data from Stage 2 runs

if aType == "XRF-S2" or aType == "KED-S2":
    # import the source data from the preprocessed output file
    spectrumIn = np.genfromtxt(outFile, delimiter=",", skip_header=3,
        skip_footer=1,
            usecols=(0,1))
    #print spectrumIn
    #print "spectrumIn=",len(spectrumIn)

    s2Bins = np.genfromtxt(outFile, delimiter=",", skip_header=3,
        skip_footer=1,
            usecols=(0))
    s2Tally = np.genfromtxt(outFile, delimiter=",", skip_header=3,
        skip_footer=1,
            usecols=(1))
    s2Error = np.genfromtxt(outFile, delimiter=",", skip_header=3,
        skip_footer=1,
            usecols=(2))

    # Convert Stage 2 bin energies to keV
    # The average offset for the Ka1 and Ka2 peaks is 0.1035 but 0.07
        should
    # make the Kb's align more closely.
    for i in range(0,len(s2Bins)):
        s2Bins[i] = s2Bins[i]*1000-0.07
```

```
    # Initialize normalized tally matrix
    s2NormTal = np.zeros((len(s2Tally),1))

    s2CdPeak = "n"
    # Search the S2 tally for the 88 keV peak
    for i in range(0,len(s2Tally)):
        if s2CdPeak == "n":
            if s2Bins[i] > 88.03: #keV
                s2CdPeak = s2Tally[i]
                #s2CdPeak = s2Tally[i] - ((s2Tally[1150]+s2Tally[1171])
                    /2)
                CdPeakLoc = i

    # Normalize the Stage 2 tally
    for i in range(0,len(s2Tally)):
        # Normalize to the Cd-109 peak
        s2NormTal[i] = s2Tally[i]/s2CdPeak
        #s2NormTal[i] = s2Tally[i]/sum(s2Tally)

# Import data from measurement files

if importSpectrum == "y":
    # Get the file name from the user and open it
    m = open(specFile,"r")
    # Import the data into a spectrum variable
    measChannel = np.genfromtxt(specFile, delimiter=",",skip_header=1,
        usecols=(0))
    measSpectrum = np.genfromtxt(specFile, delimiter=",",skip_header=1,
        usecols=(1))
    infoLen = len(measSpectrum)
```

```
spectInfo = np.genfromtxt(specFile, delimiter=",",skip_footer=
    infoLen)
# Print the energy calibration
print "\n -=- ENERGY CALIBRATION -=-"
print "Energy calibration: E(ch#)=",spectInfo[3],"+",spectInfo[4],"*
     ch#"

# Initialize the energy calibrated array
measEnergy = np.zeros((len(measChannel),1))
# Apply energy calibration to measured spectra
for i in range(0,len(measChannel)):
    measEnergy[i] = spectInfo[3]+spectInfo[4]*measChannel[i]

# Initialize the normalized spectrum array
measNormSpec = np.zeros((len(measSpectrum),1))



measCdPeak = "n"
# Search the measured data for the 88 keV peak
for i in range(0,len(measEnergy)):
    if measCdPeak == "n":
        if measEnergy[i] > 87.7: #88.03: #keV
            # Calculate the baseline
            measCdPeak = measSpectrum[i]
            #measCdPeak = measSpectrum[i] - ((measSpectrum[966]+
                measSpectrum[984])/2)
            measCdPeakLoc = i

# Normalize measured spectra for comparison
```

```
    for i in range(0,len(measSpectrum)):
        measNormSpec[i] = measSpectrum[i]/measCdPeak
        #measNormSpec[i] = measSpectrum[i]/sum(measSpectrum)
        if measNormSpec[i] == 0:
            measNormSpec[i] == 1E-50

    # Calculate measured error and normalize to Cd-109
    # Preallocate measured error array
    measError = np.zeros(len(measSpectrum))
    measFracError = np.zeros(len(measSpectrum))
    for i in range(0,len(measSpectrum)):
        measError[i] = np.sqrt(measSpectrum[i])
        if measError[i] != 0:
            measFracError[i] = measError[i]/measSpectrum[i]
        elif measError[i] == 0:
            measFracError[i] = 0
        # Convert nan's to 0


# Print message if not importing data
elif importSpectrum == "n":
    print "INFO: Measured data not imported"

# Run comparison algorithms

if importSpectrum == "y":

    # Break out if arrays are not the same length
    if len(s2NormTal) != len(measNormSpec):
        print "Stage 2 tally & measured spectra not same length!"
```

```
# Initialize residual array
modelResid = np.zeros((len(s2NormTal),1))

# Delta E arrays
delta_E_t = np.zeros((len(s2Bins),1))
delta_E_m = np.zeros((len(measEnergy),1))

# Find the delta E for both energy arrays
for i in range(1,len(s2Bins)):
    delta_E_t[i] = s2Bins[i] - s2Bins[i-1]
for i in range(1,len(measEnergy)):
    delta_E_m[i] = measEnergy[i] - measEnergy[i-1]

# Average the energies
delta_E_m = np.average(delta_E_m)
delta_E_t = np.average(delta_E_t)
print"Average measured delta E: ",delta_E_m," keV"
print"Average modeled delta E: ",delta_E_t," keV\n"

# Calculate the relative difference in corrected spectra
if aType == "KED-S2":
    ch_corr = 327 #247
elif aType == "XRF-S2":
    ch_corr = 46
# zero offset
zero_offset = 30

for i in range(0+zero_offset,len(s2NormTal)-ch_corr):
    if measNormSpec[i] == 0:
```

```
            modelResid[i] = 0 #1E-50
        else:
            modelResid[i] = abs(1-(s2NormTal[i+ch_corr]/measNormSpec[i]))
            #modelResid[i] = abs(1 - (abs(measNormSpec[i]-s2NormTal[i+
                ch_corr])/measNormSpec[i]))

    print "\n -=- WHOLE SPECTRUM ERROR ANALYSIS -=-"
    avgModelErr = np.mean(modelResid)
    print "Average model error: ", avgModelErr
    stdModelErr = np.std(modelResid)
    print "Standard deviation: ", stdModelErr

    print "\n -=- ROI SPECTRUM ERROR ANALYSIS -=-"
    if aType == "KED-S2":
        roiLower = 1507 # 114 keV
        roiUpper = 1547 # 117 keV
    elif aType == "XRF-S2":
        roiLower = 1268 # 93 keV
        roiUpper = 1679 # 121 keV

    avgModelErr = np.mean(modelResid[roiLower:roiUpper])
    print "ROI average model error: ", avgModelErr
    stdModelErr = np.std(modelResid[roiLower:roiUpper])
    print "ROI standard deviation: ",stdModelErr

    # Convert from lists to vectors
    s2Bins = np.hstack(s2Bins)

# Print message if not plotting results
elif importSpectrum == "n":
```

```
    print "INFO: Spectral comparison not run"

# K-edge continuum comparison

if aType == "KED-S2":

    print "\n-=- K-EDGE CONTINUUM ANALYSIS -=-"

    # Set measured lower and upper continuum boundaries
    k_lc_l_meas = 1215
    k_lc_u_meas = 1275
    k_uc_l_meas = 1295
    k_uc_u_meas = 1355

    # Set measured lower and upper continuum boundaries
    k_lc_l_model = 1444
    k_lc_u_model = 1517
    k_uc_l_model = 1540
    k_uc_u_model = 1612

    # Measured
    print "\n-=- MEASURED CONTINUUM BOUNDS -=-"
    print "Measured lower K-edge continuum lower bound at channel: ",
       k_lc_l_meas,"(",measEnergy[k_lc_l_meas]," keV )"
    print "Measured lower K-edge continuum upper bound at channel: ",
       k_lc_u_meas,"(",measEnergy[k_lc_u_meas]," keV )"
    print "Measured upper K-edge continuum lower bound at channel: ",
       k_uc_l_meas,"(",measEnergy[k_uc_l_meas]," keV )"
    print "Measured upper K-edge continuum upper bound at channel: ",
       k_uc_u_meas,"(",measEnergy[k_uc_u_meas]," keV )"
```

```
    # Measured
print "\n-=- MODELED CONTINUUM BOUNDS -=-"
print "Modeled lower K-edge continuum lower bound at channel: ",
    k_lc_l_model,"(",s2Bins[k_lc_l_model]," keV )"
print "Modeled lower K-edge continuum upper bound at channel: ",
    k_lc_u_model,"(",s2Bins[k_lc_u_model]," keV )"
print "Modeled upper K-edge continuum lower bound at channel: ",
    k_uc_l_model,"(",s2Bins[k_uc_l_model]," keV )"
print "Modeled upper K-edge continuum upper bound at channel: ",
    k_uc_u_model,"(",s2Bins[k_uc_u_model]," keV )"

# Determine continuum area for measured data
k_lc_sum_meas = np.sum(measNormSpec[k_lc_l_meas:k_lc_u_meas])
k_uc_sum_meas = np.sum(measNormSpec[k_uc_l_meas:k_uc_u_meas])

# Determine continuum area for modeled data
k_lc_sum_model = np.sum(s2NormTal[k_lc_l_model:k_lc_u_model])
k_uc_sum_model = np.sum(s2NormTal[k_uc_l_model:k_uc_u_model])

# Calculate continuum ratios
k_ratio_meas = k_lc_sum_meas/k_uc_sum_meas
k_ratio_model = k_lc_sum_model/k_uc_sum_model

# Calculate the uncertainty
k_ratio_meas_unc = np.average(np.average(s2Error[k_lc_l_meas:
    k_lc_u_meas])+np.average(s2Error[k_uc_l_meas:k_uc_u_meas]))
k_ratio_model_unc = np.average(np.average(s2Error[k_lc_l_model:
    k_lc_u_model])+np.average(s2Error[k_uc_l_model:k_uc_u_model]))
```

```
    # Print the results
    print "\n-=- K-EDGE ANALYSIS RESULTS -=-"
    print "Measured K-edge continuum ratio: ", k_ratio_meas
    print "Measured K-edge continuum uncertainty: ",k_ratio_meas_unc
    print "Modeled K-edge continuum ratio: ", k_ratio_model
    print "Modeled K-edge continuum uncertainty: ",k_ratio_model_unc

    print "\nK-edge Excel data format"
    print k_ratio_meas,k_ratio_meas_unc,k_ratio_model,k_ratio_model_unc

# Measured Cd-109 intensity
#if aType == "KED-S2" or aType == "XRF-S2":
#    # Calculate measured Cd-109 peak intensity
#    cd_kedge = (1/k_ratio_meas)*k_ratio_model
#    print "K-edge Cd-109 ratio: ",cd_kedge


# Uranium XRF Peak area comparison

if aType == "XRF-S2":

    print "\n-=- URANIUM K ALPHA PEAK AREA ANALYSIS -=-"

    ## Set peak boundaries
    ## U K alpha peaks
    ka2_l = 93.9
    ka2_u = 95.36
    ka1_l = 97.6
    ka1_u = 99.1
```

```
# Set measured channel variables to a string
ka2_l_chan_m = "n"
ka2_u_chan_m = "n"
ka1_l_chan_m = "n"
ka1_u_chan_m = "n"

# Measured
for i in range(0,len(measEnergy)-1):
    if (measEnergy[i] > ka2_l and ka2_l_chan_m == "n"):
        ka2_l_chan_m = i
        print "Measured Ka2 lower bound at channel: ",ka2_l_chan_m
            ,"(",measEnergy[i]," keV )"
        #print measNormSpec[i]
    elif (measEnergy[i] > ka2_u and ka2_u_chan_m == "n"):
        ka2_u_chan_m = i
        print "Measured Ka2 upper bound at channel: ",ka2_u_chan_m
            ,"(",measEnergy[i]," keV )"
        #print measNormSpec[i]
    elif (measEnergy[i] > ka1_l and ka1_l_chan_m == "n"):
        ka1_l_chan_m = i
        print "Measured Ka1 lower bound at channel: ",ka1_l_chan_m
            ,"(",measEnergy[i]," keV )"
        #print measNormSpec[i]
    elif (measEnergy[i] > ka1_u and ka1_u_chan_m == "n"):
        ka1_u_chan_m = i
        print "Measured Ka1 upper bound at channel: ",ka1_u_chan_m
            ,"(",measEnergy[i]," keV )"
        #print measNormSpec[i]

# Set tally channel variables to a string
```

```
ka2_l_chan_t = "n"
ka2_u_chan_t = "n"
ka1_l_chan_t = "n"
ka1_u_chan_t = "n"

# Tally
for i in range(0,len(s2Bins)-1):
    if (s2Bins[i] > ka2_l and ka2_l_chan_t == "n"):
        ka2_l_chan_t = i
        print "\nTally Ka2 lower bound at channel: ",ka2_l_chan_t
            ,"(",s2Bins[i]," keV )"
        #print s2NormTal[i]
    elif (s2Bins[i] > ka2_u and ka2_u_chan_t == "n"):
        ka2_u_chan_t = i
        print "Tally Ka2 upper bound at channel: ",ka2_u_chan_t,"(",
            s2Bins[i]," keV )"
        #print s2NormTal[i]
    elif (s2Bins[i] > ka1_l and ka1_l_chan_t == "n"):
        ka1_l_chan_t = i
        print "Tally Ka1 lower bound at channel: ",ka1_l_chan_t,"(",
            s2Bins[i]," keV )"
        #print s2NormTal[i]
    elif (s2Bins[i] > ka1_u and ka1_u_chan_t == "n"):
        ka1_u_chan_t = i
        print "Tally Ka1 upper bound at channel: ",ka1_u_chan_t,"(",
            s2Bins[i]," keV )"
        #print s2NormTal[i]

# Measure the peak areas
ka2_sum_meas = np.sum((measNormSpec[ka2_l_chan_m:ka2_u_chan_m]))
```

```
ka2_sum_tal = np.sum((s2NormTal[ka2_l_chan_t:ka2_u_chan_t]))

# Measure the peak areas
ka1_sum_meas = np.sum((measNormSpec[ka1_l_chan_m:ka1_u_chan_m]))
ka1_sum_tal = np.sum((s2NormTal[ka1_l_chan_t:ka1_u_chan_t]))

# Ka2 calculate the continuum
ka2_cont_m = ((measNormSpec[ka2_l_chan_m]+measNormSpec[ka2_u_chan_m
   ])/2) * (ka2_u_chan_m-ka2_l_chan_m)
ka2_cont_t = ((s2NormTal[ka2_l_chan_t]+s2NormTal[ka2_u_chan_t])/2) *
     (ka2_u_chan_t-ka2_l_chan_t)
# Ka1 calculate the continuum
ka1_cont_m = ((measNormSpec[ka1_l_chan_m]+measNormSpec[ka1_u_chan_m
   ])/2) * (ka1_u_chan_m-ka1_l_chan_m)
ka1_cont_t = ((s2NormTal[ka1_l_chan_t]+s2NormTal[ka1_u_chan_t])/2) *
     (ka1_u_chan_t-ka1_l_chan_t)

# Subtract the continuums from the peak areas
ka2_peak_m = ka2_sum_meas - ka2_cont_m
ka2_peak_t = ka2_sum_tal - ka2_cont_t
ka1_peak_m = ka1_sum_meas - ka1_cont_m
ka1_peak_t = ka1_sum_tal - ka1_cont_t

# U K beta peaks fit as doublets
kb13_l = 109.6
kb13_u = 111.8
kb24_l = 113.6
kb24_u = 116.1

# Set measured channel variables to a string
```

```
kb13_l_chan_m = "n"
kb13_u_chan_m = "n"
kb24_l_chan_m = "n"
kb24_u_chan_m = "n"

print "\n-=- URANIUM K BETA PEAK AREA ANALYSIS -=-"

# Measured
for i in range(0,len(measEnergy)-1):
    if (measEnergy[i] > kb13_l and kb13_l_chan_m == "n"):
        kb13_l_chan_m = i
        print "Measured Kb13 lower bound at channel: ",kb13_l_chan_m
            ,"(",measEnergy[i]," keV )"
        #print measNormSpec[i]
    elif (measEnergy[i] > kb13_u and kb13_u_chan_m == "n"):
        kb13_u_chan_m = i
        print "Measured Kb13 upper bound at channel: ",kb13_u_chan_m
            ,"(",measEnergy[i]," keV )"
        #print measNormSpec[i]
    elif (measEnergy[i] > kb24_l and kb24_l_chan_m == "n"):
        kb24_l_chan_m = i
        print "Measured Kb24 lower bound at channel: ",kb24_l_chan_m
            ,"(",measEnergy[i]," keV )"
        #print measNormSpec[i]
    elif (measEnergy[i] > kb24_u and kb24_u_chan_m == "n"):
        kb24_u_chan_m = i
        print "Measured Kb24 upper bound at channel: ",kb24_u_chan_m
            ,"(",measEnergy[i]," keV )"
        #print measNormSpec[i]
```

```
# Set tally channel variables to a string
kb13_l_chan_t = "n"
kb13_u_chan_t = "n"
kb24_l_chan_t = "n"
kb24_u_chan_t = "n"

# Tally
for i in range(0,len(s2Bins)-1):
    if (s2Bins[i] > kb13_l and kb13_l_chan_t == "n"):
        kb13_l_chan_t = i
        print "\nTally Kb13 lower bound at channel: ",kb13_l_chan_t
            ,"(",s2Bins[i]," keV )"
        #print s2NormTal[i]
    elif (s2Bins[i] > kb13_u and kb13_u_chan_t == "n"):
        kb13_u_chan_t = i
        print "Tally Kb13 upper bound at channel: ",kb13_u_chan_t
            ,"(",s2Bins[i]," keV )"
        #print s2NormTal[i]
    elif (s2Bins[i] > kb24_l and kb24_l_chan_t == "n"):
        kb24_l_chan_t = i
        print "Tally Kb24 lower bound at channel: ",kb24_l_chan_t
            ,"(",s2Bins[i]," keV )"
        #print s2NormTal[i]
    elif (s2Bins[i] > kb24_u and kb24_u_chan_t == "n"):
        kb24_u_chan_t = i
        print "Tally Kb24 upper bound at channel: ",kb24_u_chan_t
            ,"(",s2Bins[i]," keV )"
        #print s2NormTal[i]

# Measure the peak areas
```

```
kb13_sum_meas = np.sum((measNormSpec[kb13_l_chan_m:kb13_u_chan_m]))
kb13_sum_tal = np.sum((s2NormTal[kb13_l_chan_t:kb13_u_chan_t]))

# Measure the peak areas
kb24_sum_meas = np.sum((measNormSpec[kb24_l_chan_m:kb24_u_chan_m]))
kb24_sum_tal = np.sum((s2NormTal[kb24_l_chan_t:kb24_u_chan_t]))

# kb13 calculate the continuum
kb13_cont_m = ((measNormSpec[kb13_l_chan_m]+measNormSpec[
    kb13_u_chan_m])/2) * (kb13_u_chan_m-kb13_l_chan_m)
kb13_cont_t = ((s2NormTal[kb13_l_chan_t]+s2NormTal[kb13_u_chan_t])
    /2) * (kb13_u_chan_t-kb13_l_chan_t)
# kb24 calculate the continuum
kb24_cont_m = ((measNormSpec[kb24_l_chan_m]+measNormSpec[
    kb24_u_chan_m])/2) * (kb24_u_chan_m-kb24_l_chan_m)
kb24_cont_t = ((s2NormTal[kb24_l_chan_t]+s2NormTal[kb24_u_chan_t])
    /2) * (kb24_u_chan_t-kb24_l_chan_t)

# Subtract the continuums from the peak areas
kb13_peak_m = kb13_sum_meas - kb13_cont_m
kb13_peak_t = kb13_sum_tal - kb13_cont_t
kb24_peak_m = kb24_sum_meas - kb24_cont_m
kb24_peak_t = kb24_sum_tal - kb24_cont_t

ka1_diff = (ka1_peak_m-ka1_peak_t)/ka1_peak_m
ka2_diff = (ka2_peak_m-ka2_peak_t)/ka2_peak_m
kb13_diff = (kb13_peak_m-kb13_peak_t)/kb13_peak_m
kb24_diff = (kb24_peak_m-kb24_peak_t)/kb24_peak_m

# Calculate the peak uncertainties
```

```
# Ka2
#ka2_peak_m_unc = np.sqrt(ka2_peak_m)
#ka2_peak_t_unc = np.sqrt(ka2_peak_t)
# Ka1
#ka1_peak_m_unc = np.sqrt(ka1_peak_m)
#ka1_peak_t_unc = np.sqrt(ka1_peak_t)
# Kb13
#kb13_peak_m_unc = np.sqrt(kb13_peak_m)
#kb13_peak_t_unc = np.sqrt(kb13_peak_t)
# Kb24
#kb24_peak_m_unc = np.sqrt(kb24_peak_m)
#kb24_peak_t_unc = np.sqrt(kb24_peak_t)

# New measured spectrum error calculation
ka2_peak_m_unc = np.average(measFracError[ka2_l_chan_m:ka2_u_chan_m
    ])
ka1_peak_m_unc = np.average(measFracError[ka1_l_chan_m:ka1_u_chan_m
    ])
kb13_peak_m_unc = np.average(measFracError[kb13_l_chan_m:
    kb13_u_chan_m])
kb24_peak_m_unc = np.average(measFracError[kb24_l_chan_m:
    kb24_u_chan_m])

# New tally error calculation
ka2_peak_t_unc = np.average(s2Error[ka2_l_chan_t:ka2_u_chan_t])
ka1_peak_t_unc = np.average(s2Error[ka1_l_chan_t:ka1_u_chan_t])
kb13_peak_t_unc = np.average(s2Error[kb13_l_chan_t:kb13_u_chan_t])
kb24_peak_t_unc = np.average(s2Error[kb24_l_chan_t:kb24_u_chan_t])

print "\n-=- URANIUM PEAK AREAS -=-"
```

```
print "Ka2 measured: ",ka2_peak_m
print "Ka2 measured uncertainty: ",ka2_peak_m_unc
print "Ka2 tally: ",ka2_peak_t
print "Ka2 tally uncertainty: ",ka2_peak_t_unc

print "Ka1 measured: ",ka1_peak_m
print "Ka1 measured uncertainty: ",ka1_peak_m_unc
print "Ka1 tally: ",ka1_peak_t
print "Ka1 tally uncertainty: ",ka1_peak_t_unc

print "\nKb13 measured: ",kb13_peak_m
print "Kb13 measured uncertainty: ",kb13_peak_m_unc
print "Kb13 tally: ",kb13_peak_t
print "Kb13 tally uncertainty: ",kb13_peak_t_unc

print "\nKb24 measured: ",kb24_peak_m
print "Kb24 measured uncertainty: ",kb24_peak_m_unc
print "Kb24 tally: ",kb24_peak_t
print "Kb24 tally uncertainty: ",kb24_peak_t_unc

print "\nKa1 error: ",ka1_diff*100," %"
print "Ka2 error: ",ka2_diff*100," %"
print "Kb13 error: ",kb13_diff*100," %"
print "Kb24 error: ",kb24_diff*100," %"
print "\n"

# Excel data format
print "\nUranium Excel data format\n"
```

```
    #print ka1_peak_m+","+ka1_peak_m_unc+","+ka2_peak_m+","+
        ka2_peak_m_unc+","+kb13_peak_m+","+kb13_peak_m_unc+","+
        kb24_peak_m+","+kb24_peak_m_unc+","+ka1_peak_t+","+ka1_peak_t_unc
        +","+ka2_peak_t+","+ka2_peak_t_unc+","+kb13_peak_t+","+
        kb13_peak_t_unc+","+kb24_peak_t+","+kb24_peak_t_unc
    print ka1_peak_m,ka1_peak_m_unc,ka2_peak_m,ka2_peak_m_unc,
        kb13_peak_m,kb13_peak_m_unc,kb24_peak_m,kb24_peak_m_unc,
        ka1_peak_t,ka1_peak_t_unc,ka2_peak_t,ka2_peak_t_unc,kb13_peak_t,
        kb13_peak_t_unc,kb24_peak_t,kb24_peak_t_unc

# Plutionum XRF peaks

if aType == "XRF-S2" and sType == "U-Pu":

    print "\n-=- PLUTIONIUM K ALPHA PEAK AREA ANALYSIS -=-"

    # Set peak boundaries
    # Pu K alpha peaks
    ka2_l = 98.9
    ka2_u = 100.1
    ka1_l = 97.8
    ka1_u = 99.0

    # Set measured channel variables to a string
    ka2_l_chan_m = "n"
    ka2_u_chan_m = "n"
    ka1_l_chan_m = "n"
    ka1_u_chan_m = "n"

    # Measured
```

```
for i in range(0,len(measEnergy)-1):
    if (measEnergy[i] > ka2_l and ka2_l_chan_m == "n"):
        ka2_l_chan_m = i
        print "Measured Ka2 lower bound at channel: ",ka2_l_chan_m
            ,"(",measEnergy[i]," keV )"
        #print measNormSpec[i]
    elif (measEnergy[i] > ka2_u and ka2_u_chan_m == "n"):
        ka2_u_chan_m = i
        print "Measured Ka2 upper bound at channel: ",ka2_u_chan_m
            ,"(",measEnergy[i]," keV )"
        #print measNormSpec[i]
    elif (measEnergy[i] > ka1_l and ka1_l_chan_m == "n"):
        ka1_l_chan_m = i
        print "Measured Ka1 lower bound at channel: ",ka1_l_chan_m
            ,"(",measEnergy[i]," keV )"
        #print measNormSpec[i]
    elif (measEnergy[i] > ka1_u and ka1_u_chan_m == "n"):
        ka1_u_chan_m = i
        print "Measured Ka1 upper bound at channel: ",ka1_u_chan_m
            ,"(",measEnergy[i]," keV )"
        #print measNormSpec[i]

# Set tally channel variables to a string
ka2_l_chan_t = "n"
ka2_u_chan_t = "n"
ka1_l_chan_t = "n"
ka1_u_chan_t = "n"

# Tally
for i in range(0,len(s2Bins)-1):
```

```
    if (s2Bins[i] > ka2_l and ka2_l_chan_t == "n"):
        ka2_l_chan_t = i
        print "\nTally Ka2 lower bound at channel: ",ka2_l_chan_t
            ,"(",s2Bins[i]," keV )"
        #print s2NormTal[i]
    elif (s2Bins[i] > ka2_u and ka2_u_chan_t == "n"):
        ka2_u_chan_t = i
        print "Tally Ka2 upper bound at channel: ",ka2_u_chan_t,"(",
            s2Bins[i]," keV )"
        #print s2NormTal[i]
    elif (s2Bins[i] > ka1_l and ka1_l_chan_t == "n"):
        ka1_l_chan_t = i
        print "Tally Ka1 lower bound at channel: ",ka1_l_chan_t,"(",
            s2Bins[i]," keV )"
        #print s2NormTal[i]
    elif (s2Bins[i] > ka1_u and ka1_u_chan_t == "n"):
        ka1_u_chan_t = i
        print "Tally Ka1 upper bound at channel: ",ka1_u_chan_t,"(",
            s2Bins[i]," keV )"
        #print s2NormTal[i]

# Measure the peak areas
ka2_sum_meas = np.sum((measNormSpec[ka2_l_chan_m:ka2_u_chan_m]))
ka2_sum_tal = np.sum((s2NormTal[ka2_l_chan_t:ka2_u_chan_t]))

# Measure the peak areas
ka1_sum_meas = np.sum((measNormSpec[ka1_l_chan_m:ka1_u_chan_m]))
ka1_sum_tal = np.sum((s2NormTal[ka1_l_chan_t:ka1_u_chan_t]))

# Ka2 calculate the continuum
```

```
ka2_cont_m = ((measNormSpec[ka2_l_chan_m]+measNormSpec[ka2_u_chan_m
    ])/2) * (ka2_u_chan_m-ka2_l_chan_m)
ka2_cont_t = ((s2NormTal[ka2_l_chan_t]+s2NormTal[ka2_u_chan_t])/2) *
     (ka2_u_chan_t-ka2_l_chan_t)
# Ka1 calculate the continuum
ka1_cont_m = ((measNormSpec[ka1_l_chan_m]+measNormSpec[ka1_u_chan_m
    ])/2) * (ka1_u_chan_m-ka1_l_chan_m)
ka1_cont_t = ((s2NormTal[ka1_l_chan_t]+s2NormTal[ka1_u_chan_t])/2) *
     (ka1_u_chan_t-ka1_l_chan_t)

# Subtract the continuums from the peak areas
ka2_peak_m = ka2_sum_meas - ka2_cont_m
ka2_peak_t = ka2_sum_tal - ka2_cont_t
ka1_peak_m = ka1_sum_meas - ka1_cont_m
ka1_peak_t = ka1_sum_tal - ka1_cont_t

if ka2_peak_m < 0:
    ka2_cont_m = 0
    ka2_peak_m = 0 #ka2_sum_meas - ka2_cont_m
if ka2_peak_t < 0:
    ka2_cont_t = 0
    ka2_peak_t = 0 #ka2_sum_tal - ka2_cont_t
if ka1_peak_m < 0:
    ka2_cont_m = 0
    ka1_peak_m = 0 #ka1_sum_meas - ka1_cont_m
if ka1_peak_t < 0:
    ka1_cont_t = 0
    ka1_peak_t = 0 #ka1_sum_tal - ka1_cont_t

# Pu K beta peaks fit as doublets
```

```
kb13_l = 110.0
kb13_u = 111.6
kb24_l = 114.0
kb24_u = 116.1

# Set measured channel variables to a string
kb13_l_chan_m = "n"
kb13_u_chan_m = "n"
kb24_l_chan_m = "n"
kb24_u_chan_m = "n"

print "\n-=- PLUTONIUM K BETA PEAK AREA ANALYSIS -=-"

# Measured
for i in range(0,len(measEnergy)-1):
    if (measEnergy[i] > kb13_l and kb13_l_chan_m == "n"):
        kb13_l_chan_m = i
        print "Measured Kb13 lower bound at channel: ",kb13_l_chan_m
            ,"(",measEnergy[i]," keV )"
        #print measNormSpec[i]
    elif (measEnergy[i] > kb13_u and kb13_u_chan_m == "n"):
        kb13_u_chan_m = i
        print "Measured Kb13 upper bound at channel: ",kb13_u_chan_m
            ,"(",measEnergy[i]," keV )"
        #print measNormSpec[i]
    elif (measEnergy[i] > kb24_l and kb24_l_chan_m == "n"):
        kb24_l_chan_m = i
        print "Measured Kb24 lower bound at channel: ",kb24_l_chan_m
            ,"(",measEnergy[i]," keV )"
        #print measNormSpec[i]
```

```
    elif (measEnergy[i] > kb24_u and kb24_u_chan_m == "n"):
        kb24_u_chan_m = i
        print "Measured Kb24 upper bound at channel: ",kb24_u_chan_m
            ,"(",measEnergy[i]," keV )"
        #print measNormSpec[i]

# Set tally channel variables to a string
kb13_l_chan_t = "n"
kb13_u_chan_t = "n"
kb24_l_chan_t = "n"
kb24_u_chan_t = "n"

# Tally
for i in range(0,len(s2Bins)-1):
    if (s2Bins[i] > kb13_l and kb13_l_chan_t == "n"):
        kb13_l_chan_t = i
        print "\nTally Kb13 lower bound at channel: ",kb13_l_chan_t
            ,"(",s2Bins[i]," keV )"
        #print s2NormTal[i]
    elif (s2Bins[i] > kb13_u and kb13_u_chan_t == "n"):
        kb13_u_chan_t = i
        print "Tally Kb13 upper bound at channel: ",kb13_u_chan_t
            ,"(",s2Bins[i]," keV )"
        #print s2NormTal[i]
    elif (s2Bins[i] > kb24_l and kb24_l_chan_t == "n"):
        kb24_l_chan_t = i
        print "Tally Kb24 lower bound at channel: ",kb24_l_chan_t
            ,"(",s2Bins[i]," keV )"
        #print s2NormTal[i]
    elif (s2Bins[i] > kb24_u and kb24_u_chan_t == "n"):
```

```
        kb24_u_chan_t = i
        print "Tally Kb24 upper bound at channel: ",kb24_u_chan_t
            ,"(",s2Bins[i]," keV )"
        #print s2NormTal[i]

# Measure the peak areas
kb13_sum_meas = np.sum((measNormSpec[kb13_l_chan_m:kb13_u_chan_m]))
kb13_sum_tal = np.sum((s2NormTal[kb13_l_chan_t:kb13_u_chan_t]))

# Measure the peak areas
kb24_sum_meas = np.sum((measNormSpec[kb24_l_chan_m:kb24_u_chan_m]))
kb24_sum_tal = np.sum((s2NormTal[kb24_l_chan_t:kb24_u_chan_t]))

# kb13 calculate the continuum
kb13_cont_m = ((measNormSpec[kb13_l_chan_m]+measNormSpec[
    kb13_u_chan_m])/2) * (kb13_u_chan_m-kb13_l_chan_m)
kb13_cont_t = ((s2NormTal[kb13_l_chan_t]+s2NormTal[kb13_u_chan_t])
    /2) * (kb13_u_chan_t-kb13_l_chan_t)
# kb24 calculate the continuum
kb24_cont_m = ((measNormSpec[kb24_l_chan_m]+measNormSpec[
    kb24_u_chan_m])/2) * (kb24_u_chan_m-kb24_l_chan_m)
kb24_cont_t = ((s2NormTal[kb24_l_chan_t]+s2NormTal[kb24_u_chan_t])
    /2) * (kb24_u_chan_t-kb24_l_chan_t)


# Subtract the continuums from the peak areas
kb13_peak_m = kb13_sum_meas - kb13_cont_m
kb13_peak_t = kb13_sum_tal - kb13_cont_t
kb24_peak_m = kb24_sum_meas - kb24_cont_m
kb24_peak_t = kb24_sum_tal - kb24_cont_t
```

```
if kb13_peak_m < 0:
    kb13_cont_m = 0
    kb13_peak_m = 0 #ka2_sum_meas - ka2_cont_m
if kb13_peak_t < 0:
    kb13_cont_t = 0
    kb13_peak_t = 0 #ka2_sum_tal - ka2_cont_t
if kb24_peak_m < 0:
    kb24_cont_m = 0
    kb24_peak_m = 0 #ka1_sum_meas - ka1_cont_m
if ka1_peak_t < 0:
    kb24_cont_t = 0
    kb24_peak_t = 0 #ka1_sum_tal - ka1_cont_t

# Calculate the peak uncertainties
# Ka2
#ka2_peak_m_unc = np.sqrt(ka2_peak_m)
#ka2_peak_t_unc = np.sqrt(ka2_peak_t)
# Ka1
#ka1_peak_m_unc = np.sqrt(ka1_peak_m)
#ka1_peak_t_unc = np.sqrt(ka1_peak_t)
# Kb13
#kb13_peak_m_unc = np.sqrt(kb13_peak_m)
#kb13_peak_t_unc = np.sqrt(kb13_peak_t)
# Kb24
#kb24_peak_m_unc = np.sqrt(kb24_peak_m)
#kb24_peak_t_unc = np.sqrt(kb24_peak_t)

# New measured spectrum error calculation
```

```
ka2_peak_m_unc = np.average(measFracError[ka2_l_chan_m:ka2_u_chan_m
    ])
ka1_peak_m_unc = np.average(measFracError[ka1_l_chan_m:ka1_u_chan_m
    ])
kb13_peak_m_unc = np.average(measFracError[kb13_l_chan_m:
    kb13_u_chan_m])
kb24_peak_m_unc = np.average(measFracError[kb24_l_chan_m:
    kb24_u_chan_m])

# New tally error calculation
ka2_peak_t_unc = np.average(s2Error[ka2_l_chan_t:ka2_u_chan_t])
ka1_peak_t_unc = np.average(s2Error[ka1_l_chan_t:ka1_u_chan_t])
kb13_peak_t_unc = np.average(s2Error[kb13_l_chan_t:kb13_u_chan_t])
kb24_peak_t_unc = np.average(s2Error[kb24_l_chan_t:kb24_u_chan_t])

#ka1_diff = (ka1_peak_m-ka1_peak_t)/ka1_peak_m
#ka2_diff = (ka2_peak_m-ka2_peak_t)/ka2_peak_m
#kb13_diff = (kb13_peak_m-kb13_peak_t)/kb13_peak_m
#kb24_diff = (kb24_peak_m-kb24_peak_t)/kb24_peak_m

print "\n-=- PLUTONIUM PEAK AREAS -=-"

print "Ka2 measured: ",ka2_peak_m
print "Ka2 measured uncertainty: ",ka2_peak_m_unc
print "Ka2 tally: ",ka2_peak_t
print "Ka2 tally uncertainty: ",ka2_peak_t_unc

print "Ka1 measured: ",ka1_peak_m
print "Ka1 measured uncertainty: ",ka1_peak_m_unc
print "Ka1 tally: ",ka1_peak_t
```

```
print "Ka1 tally uncertainty: ",ka1_peak_t_unc

print "\nKb13 measured: ",kb13_peak_m
print "Kb13 measured uncertainty: ",kb13_peak_m_unc
print "Kb13 tally: ",kb13_peak_t
print "Kb13 tally uncertainty: ",kb13_peak_t_unc

print "\nKb24 measured: ",kb24_peak_m
print "Kb24 measured uncertainty: ",kb24_peak_m_unc
print "Kb24 tally: ",kb24_peak_t
print "Kb24 tally uncertainty: ",kb24_peak_t_unc

#print "\nKa1 error: ",ka1_diff*100," %"
#print "Ka2 error: ",ka2_diff*100," %"
#print "Kb13 error: ",kb13_diff*100," %"
#print "Kb24 error: ",kb24_diff*100," %"
#print "\n"

# Excel data format
print "\nPlutonium Excel data format\n"
#print ka1_peak_m+","+ka1_peak_m_unc+","+ka2_peak_m+","+
   ka2_peak_m_unc+","+kb13_peak_m+","+kb13_peak_m_unc+","+
   kb24_peak_m+","+kb24_peak_m_unc+","+ka1_peak_t+","+ka1_peak_t_unc
   +","+ka2_peak_t+","+ka2_peak_t_unc+","+kb13_peak_t+","+
   kb13_peak_t_unc+","+kb24_peak_t+","+kb24_peak_t_unc
print ka1_peak_m,ka1_peak_m_unc,ka2_peak_m,ka2_peak_m_unc,
   kb13_peak_m,kb13_peak_m_unc,kb24_peak_m,kb24_peak_m_unc,
   ka1_peak_t,ka1_peak_t_unc,ka2_peak_t,ka2_peak_t_unc,kb13_peak_t,
   kb13_peak_t_unc,kb24_peak_t,kb24_peak_t_unc
```

```
# Plot the results

chan = range(0,2048)

# correct the residuals bins by 4 keV
if aType == "KED-S2":
    s2ResidBins = s2Bins[:] - 4
elif aType == "XRF-S2":
    s2ResidBins = s2Bins[:] + 3

print "\n-=- PLOTTING -=-"
# If plotResults is "y" then call matplotlib
if plotResults == "y":

    # close any previous figures
    plt.close("all")

    # Prompt for plot title
    plotTitle = "" #raw_input("Enter string for plot title: ")

    # Set the figure dimensions
    plt.figure(figsize=(8,4))

    # Plot the tally results
    plt.subplot(2,1,1)
    plt.ylabel("Relative Intensity")
    plt.xlim((0,155))
    plt.ylim((1E-4,1E2))
```

```
plt.semilogy(s2Bins,s2NormTal,color="red",label="MCNP")

# Set the title
plt.title(plotTitle)

# Plot the measured spectrum
plt.subplot(2,1,1)
plt.semilogy(measEnergy,measNormSpec,color="blue",label="
    Experimental")

#plt.xlim((0,155))
plt.ylabel("Relative Intensity", fontsize=10)
# Set the legend location
plt.legend(loc=1,prop={'size':6})
plt.tick_params(labelsize=12)
plt.tick_params(which='both', width=1, labelsize=10)
plt.tick_params(which='major', length=8)
plt.grid(b=True,which="major")
if aType == "XRF-S2":
    if sType == "U":
        # U case
        plt.axvspan(93,100,facecolor='0.5',alpha=0.25)
        plt.axvspan(108,117,facecolor='0.5',alpha=0.25)
    elif sType == "U-Pu":
        # U-Pu case
        plt.axvspan(93,100,facecolor='0.5',alpha=0.25)
        plt.axvspan(108,118,facecolor='0.5',alpha=0.25)
        plt.axvspan(93,99.09,facecolor='0.5',alpha=0.25)
        plt.axvspan(108,116,facecolor='0.5',alpha=0.25)
        plt.axvspan(99.1,100,facecolor='g',alpha=0.25)
```

```
        plt.axvspan(103,104.5,facecolor='g',alpha=0.25)
        plt.axvspan(116.5,118,facecolor='g',alpha=0.25)
        plt.axvspan(119.5,121.5,facecolor='g',alpha=0.25)
elif aType == "KED-S2":
    plt.axvspan(114,117,facecolor='0.5',alpha=0.25)


# Plot the residuals
plt.subplot(2,1,2)
plt.scatter(s2ResidBins,modelResid, marker='.',s=5)
plt.xlabel("Energy (keV)", fontsize=10)
plt.xlim((0,155))
#plt.ylim((10E-5,10E2))
if aType == "KED-S2":
    plt.ylim((0,10))
elif aType == "XRF-S2":
    plt.ylim((0,10))
#plt.yscale("log")  #plt.yscale("log")
plt.ylabel("Abs. Fractional Difference", fontsize=10)
plt.tick_params(labelsize=12)
plt.tick_params(which='both', width=1,labelsize=10)
plt.tick_params(which='major', length=8)
plt.grid(b=True,which="major")
if aType == "XRF-S2":
    if sType == "U":
        # U case
        plt.axvspan(93,100,facecolor='0.5',alpha=0.25)
        plt.axvspan(108,117,facecolor='0.5',alpha=0.25)
    elif sType == "U-Pu":
        # U-Pu case
```

```
            plt.axvspan(93,100,facecolor='0.5',alpha=0.25)
            plt.axvspan(108,118,facecolor='0.5',alpha=0.25)
            plt.axvspan(93,99.09,facecolor='0.5',alpha=0.25)
            plt.axvspan(108,116,facecolor='0.5',alpha=0.25)
            plt.axvspan(99.1,100,facecolor='g',alpha=0.25)
            plt.axvspan(103,104.5,facecolor='g',alpha=0.25)
            plt.axvspan(116.5,118,facecolor='g',alpha=0.25)
            plt.axvspan(119.5,121.5,facecolor='g',alpha=0.25)
    elif aType == "KED-S2":
        plt.axvspan(114,117,facecolor='0.5',alpha=0.25)

    # Ask user to save the file
    #saveFile = "y" #raw_input("Do you want to save full figure as a PDF
        ? (y/n): ")
    if saveFile == "y":
        #figName = raw_input("Enter file name: ")
        plt.savefig(figName, dpi=1000, format='pdf', orientation='
            landscape',
            bbox_inches='tight')
        plt.savefig(figName2, dpi=1000, format='png', orientation='
            landscape',
            bbox_inches='tight')
        print "INFO: Full figures saved"
    elif saveFile =="n":
        print "INFO: Full figures not saved"

#######
# Plot detailed version
# Set the figure dimensions
    plt.figure(figsize=(8,4))
```

```
# Plot the tally results
plt.subplot(2,1,1)
plt.ylabel("Relative Intensity")
plt.ylim((1E-4,1E2))
plt.semilogy(s2Bins,s2NormTal,color="red",label="MCNP")
#plt.errorbar(s2Bins,s2Tally,yerr=corrError*s2Tally,color="red",
#    label="MCNP",errorevery=10)
## Set the title
plt.title(plotTitle)

# Plot the measured spectrum
plt.subplot(2,1,1)
plt.semilogy(measEnergy,measNormSpec,color="blue",label="
    Experimental")
plt.xlim((92,122))
plt.ylabel("Relative Intensity", fontsize=10)
# Set the legend location
plt.legend(loc=1,prop={'size':6})
plt.tick_params(labelsize=12)
plt.tick_params(which='both', width=1, labelsize=10)
plt.tick_params(which='major', length=8)
plt.grid(b=True,which="major")
if aType == "XRF-S2":
    if sType == "U":
        # U case
        plt.axvspan(93,100,facecolor='0.5',alpha=0.25)
        plt.axvspan(108,117,facecolor='0.5',alpha=0.25)
    elif sType == "U-Pu":
        # U-Pu case
```

```
        plt.axvspan(93,100,facecolor='0.5',alpha=0.25)
        plt.axvspan(108,118,facecolor='0.5',alpha=0.25)
        plt.axvspan(93,99.09,facecolor='0.5',alpha=0.25)
        plt.axvspan(108,116,facecolor='0.5',alpha=0.25)
        plt.axvspan(99.1,100,facecolor='g',alpha=0.25)
        plt.axvspan(103,104.5,facecolor='g',alpha=0.25)
        plt.axvspan(116.5,118,facecolor='g',alpha=0.25)
        plt.axvspan(119.5,121.5,facecolor='g',alpha=0.25)
elif aType == "KED-S2":
    plt.axvspan(114,117,facecolor='0.5',alpha=0.25)

# Plot the residuals
plt.subplot(2,1,2)
plt.scatter(s2ResidBins,modelResid,marker='.',s=5)
plt.xlabel("Energy (keV)", fontsize=10)
plt.xlim((92,122))
#plt.ylim((10E-5,10E2))
if aType == "KED-S2":
    plt.ylim((0,2))
elif aType == "XRF-S2":
    plt.ylim((0,5))
#plt.yscale("log")
plt.ylabel("Abs. Fractional Difference", fontsize=10)
plt.tick_params(labelsize=12)
plt.tick_params(which='both', width=1,labelsize=10)
plt.tick_params(which='major', length=8)
plt.grid(b=True,which="major")
if aType == "XRF-S2":
    if sType == "U":
        # U case
```

```
            plt.axvspan(93,100,facecolor='0.5',alpha=0.25)
            plt.axvspan(108,117,facecolor='0.5',alpha=0.25)
        elif sType == "U-Pu":
            # U-Pu case
            plt.axvspan(93,100,facecolor='0.5',alpha=0.25)
            plt.axvspan(108,118,facecolor='0.5',alpha=0.25)
            plt.axvspan(93,99.09,facecolor='0.5',alpha=0.25)
            plt.axvspan(108,116,facecolor='0.5',alpha=0.25)
            plt.axvspan(99.1,100,facecolor='g',alpha=0.25)
            plt.axvspan(103,104.5,facecolor='g',alpha=0.25)
            plt.axvspan(116.5,118,facecolor='g',alpha=0.25)
            plt.axvspan(119.5,121.5,facecolor='g',alpha=0.25)
    elif aType == "KED-S2":
        plt.axvspan(114,117,facecolor='0.5',alpha=0.25)

    # Ask user to save the file
    #saveFile = "y" #raw_input("Do you want to save zoomed figure as a
        PDF? (y/n): ")
    if saveFile == "y":
        #figNameROI = raw_input("Enter file name: ")
        plt.savefig(figNameROI, dpi=1000, format='pdf', orientation='
            landscape',
            bbox_inches='tight')
        plt.savefig(figNameROI2, dpi=1000, format='png', orientation='
            landscape',
            bbox_inches='tight')
        print "INFO: Detailed figures saved"
    elif saveFile =="n":
        print "INFO: Detailed figures not saved"
#######
```

```
    #newchan=np.zeros((len(chan),1))
    #
    #for i in range(0,len(chan)):
    #    newchan[i] = chan[i]+0
    #plt.figure(figsize=(8,4))
    #plt.subplot(1,1,1)
    #plt.semilogy(chan,s2NormTal,color='red')
    #plt.semilogy(newchan,measNormSpec,color='blue')
    #plt.grid(b=True,which="major")
    #plt.xlim((0,2050))
    ###plt.subplot(2,1,2)
    ##plt.xlim((0,2000))
    ##plt.yscale("log")
    ##plt.ylim((10E-6,10E3))
    ##plt.grid(b=True,which="major")
    ##plt.scatter(chan,modelResid,marker='.',s=5)

    # show the plots

## Print message if not plotting results
elif plotResults == "n":
    print "INFO: Results not plotted"

plt.show()



# close all open files
f.close()
o.close()
```

```
if stage == "s2":
    r.close()
#s.close()
```

### C.1.2 Pyrochemical Samples

```
# HPAT: HKED Python Analysis Tool Pyro
# Version 2.0
# Matthew T. Cook
# 12 March 2014
# Department of Nuclear Engineering
# University of Tennessee, Knoxville

# import modules
import os as os
import numpy as np
import matplotlib.pyplot as plt
import warnings



# Warning control
# Set script to ignore runtime warnings from modelResid divide by zero
warnings.simplefilter("ignore")



# Clear the screen
os.system('clear')

# Identify script
print " ---------------------------------"
```

```
print "| HKED Python Analysis Tool v. 2.0|"
print " --------------------------------"

# Run in automatic
auto = "n" # "y" or "n"

# Save plot files?
saveFile = "y" # "y" or "n"

plotResults = "y"

# Concentration
conc = "LBC"
run  = "ked"  # "xrf" or "ked"
#sampType = "u" # "u" or "upu"
stage = "s2"  # "s1" or "s2"
# Hardcode sample type
sType = "U-Pu"  # Options: "U" or "U-Pu"
cat = "aqueous"

if stage == "s1":
    outdir = "s1_cases"
elif stage == "s2":
    outdir = "s2_cases"

# Hardcode file name in for now
specFile = "data/"+run+"_"+conc+"gL_"+sType+".txt"
#specFile = "data/old/300gL_ked_u.txt"
```

```
fileName = outdir+"/"+run+"/"+cat+"/hked_v11_"+run+"_"+stage+"_"+conc+"
    gL_"+sType+".inp.o"

fileName = "s2_cases/ked/pyro/hked_v12_ked_s2_mk4_er.inp.o"



name = conc+"gL_"+run+"_"+sType
name = "mk4_er_ked"

# Hardcode figure file names
figName = name + "_full.pdf"
figNameROI = name + "_detail.pdf"
figName2 = name + "_full.png"
figNameROI2 = name + "_detail.png"



# Open and create files needed by this script
# Open the input file for reading
f = open(fileName, "r")

# rename the input file and open it as a post processed output
base = os.path.splitext(fileName)[0]
outFile = base + ".opp"
sourceFile = base + ".src"
if stage == "s2":
    resultsFile = base + ".res"
    r = open(resultsFile, "w")
    r.write(name+",")

# create and open the output files for writing
```

```
o = open(outFile, "w")
#s = open(sourceFile, "wb")


# Determine the run type and search terms

# Tell the user what's going on
print "\nProcessing MCNP output:", fileName
# Search the file for XRF-RUN or KED-RUN tags
with f as search:
    for line in search:
        # Remove '\n' at end of line
        line = line.rstrip()
        if "xrf-s1" in line:
            aType = "XRF-S1"
            print "Run Type:", aType
            break
        elif "xrf-s2" in line:
            aType = "XRF-S2"
            print "Run Type:", aType
            break
        elif "ked-s1" in line:
            aType = "KED-S1"
            print "Run Type:", aType
            break
        elif "ked-s2" in line:
            aType = "KED-S2"
            print "Run Type:", aType
            break
```

```
if aType == "XRF-S2" or aType == "KED-S2":
    # Ask the user what to do
    #plotResults = "n" #raw_input("Plot the results? (y/n): ")
    #plotResults = "n"

    # Ask user if a measured spectrum is to be analyzed
    importSpectrum = "y" #raw_input("Analyze a measured spectrum? (y/n):
        ")
if aType == "XRF-S1" or aType == "KED-S1":
    importSpectrum = "n"
    plotResults = "n"



# Tell the user where the extracted spectrum is
print "MCNP tally extracted to:",outFile

# Set the tally surface/volume search string
if aType == "XRF-S1":
    searchPhrase = " surface 609.2"
    print "Searched MCNP output for:", searchPhrase
elif aType == "XRF-S2":
    searchPhrase = " cell 75"
    print "Searched MCNP output for:", searchPhrase
elif aType == "KED-S1":
    searchPhrase = " surface 313.2"
    print "Searched MCNP output for:", searchPhrase
elif aType == "KED-S2":
    searchPhrase = " cell 44"
    print "Searched MCNP output for:", searchPhrase
```

```
# Create an output file and extract spectrum from MCNP output

if aType == "XRF-S1" or aType == "KED-S1":
    # Identify Stage 1 analysis
    print "Extracting Stage 1 F2 tally..."
    # Search for the specified string in the input file
    with open(fileName) as search:
        for line in search:
            # Remove '\n' at end of line
            line = line.rstrip()
            if searchPhrase in line:
                #for x in range (0,2053):
                #for x in range (0,602):
                #for x in range (0,2051):
                for x in range (0,8194):
                        line = search.next()
                        # Remove the spaces between tally values
                        # and replace with commas
                        line = line.replace('  ','')
                        line = line.replace(' ',',')
                        line = line.replace(' ',',')
                        line = line.replace(',,energy,','energy')
                        line = line.replace(',,total,,',',total,')
                        # Write the data to analysis file in CSV format
                        o.write(line)
    # Close and reopen output file to flush the buffer
    o.close()
    o = open(outFile,"r")
```

```
# Create source file from Stage 1 tally for Stage 2 source


# write the Stage 2 SDEF options to the new source file with a Cd-109
    check source
if aType == "XRF-S1":
    s = open(sourceFile, "wb")
    s2sdef = "SDEF VEC=-0.692 -0.721 0 DIR=1 POS=D1 ERG=FPOS=D2 PAR=2
        ARA=0.001 \n"
    s2sdef = s2sdef + "SI1 L -9.57 -9.97 0   -9.57 -9.97 0 \n"
    s2sdef = s2sdef + "SP1   1               1E-3 \n"
    s2sdef = s2sdef + "DS2 S 4               3 \n"
    s2sdef = s2sdef + "SI3 L 0.02199 0.022163 0.024912 0.02943 0.025455
        0.0880336 \n"
    s2sdef = s2sdef + "SP3 D 0.298  0.561   0.048   0.092  0.0231
        0.0370"
elif aType =="KED-S1":
    s = open(sourceFile, "wb")
    s2sdef = "SDEF VEC=1 0 0 DIR=1 POS=D1 ERG=FPOS=D2 PAR=2 ARA=0.001 \n
        "
    s2sdef = s2sdef + "SI1 L 19.8545 0 0   19.8545 1 0 \n"
    s2sdef = s2sdef + "SP1   1               2E-1 \n"
    s2sdef = s2sdef + "DS2 S 4               3 \n"
    s2sdef = s2sdef + "SI3 L 0.02199 0.022163 0.024912 0.02943 0.025455
        0.0880336 \n"
    s2sdef = s2sdef + "SP3 D 0.298  0.561   0.048   0.092  0.0231
        0.0370"


if aType == "XRF-S1" or aType == "KED-S1":
    print "Writing Stage 2 source file..."
```

```
# write source file header
sheader = "c Source file processed from output: "
s.write(sheader + fileName + "\n")
s.write(s2sdef + "\n")

# import the source data from the preprocessed output file
sourcein = np.genfromtxt(outFile, delimiter=",", skip_header=1,
    skip_footer=1,
        usecols=(0,1))
mcnpEnergy = np.genfromtxt(outFile, delimiter=",", skip_header=1,
    skip_footer=1,
        usecols=(0))
mcnpTally = np.genfromtxt(outFile, delimiter=",", skip_header=1,
    skip_footer=1,
        usecols=(1))
mcnpError= np.genfromtxt(outFile, delimiter=",", skip_header=1,
    skip_footer=1,
        usecols=(2))

# sum the source and copy the energies
sumsource = sum(sourcein[:,1])
sourceenergy = sourcein[:,0]

# normalize the Stage 1 tally
for x in range(0,len(sourcein)):
    sourcenorm = sourcein[:,1]/sumsource

# Polynomial fit parameters
#p1  =   5.347e-06
#p2  =   5.042e-06
```

```
#p3  =  -3.899e-05
#p4  =  -3.092e-05
#p5  =   8.373e-05
#p6  =   5.541e-05
#p7  =  -5.538e-05
#p8  =   -5.86e-05
#p9  =  -0.0002095
#p10 = -0.0002799
p1 = -0.2099
p2 =  0.07759
p3 = -0.0003616
p4 = -2.786E-5

# Correct the energy
for i in range(0,len(sourceenergy)):
    sourceenergy[i] = sourceenergy[i] - (p1*sourceenergy[i]**3 + p2*
        sourceenergy[i]**2 + p3*sourceenergy[i] + p4)

# write source energies
for x in range(0,len(sourcein)):
    if x == 0:
        #soutline1 = "SI1 L "
        # Use SI1 L for source with Cd-109
        soutline1 = "SI4 L "
        soutline2 = str(sourceenergy[x])
        s.write(soutline1)
        s.write(soutline2 + "\n")
    else:
        soutline = ("     " + str(sourceenergy[x]))
        s.write(soutline + "\n")
```

```
    # write the source intensities
    for x in range(0,len(sourcein)):
        if x == 0:
            #soutline1 = "SP1 D "
            # Use SP4 D for source with Cd-109
            soutline1 = "SP4 D "
            soutline2 = str(sourcenorm[x])
            s.write(soutline1)
            s.write(soutline2 + "\n")
        else:
            soutline = ("     " + str(sourcenorm[x]))
            s.write(soutline + "\n")

    # tell user where the source file was written
    print "Stage 1 tally written to Stage 2 source file:", sourceFile
    s.close()

    errorin = np.genfromtxt(outFile, delimiter=",", skip_header=1,
        skip_footer=1,
            usecols=(0,2))


# Import data from Stage 2 runs

if aType == "XRF-S2" or aType == "KED-S2":
    print "Extracting Stage 2 F8 tally..."
    with open(fileName) as search:
        for line in search:
            line = line.rstrip() # remove '\n' at end of line
```

```
            if searchPhrase in line:
                for x in range (0,2052):
                    line = search.next()
                    # remove the spaces between tally values
                    # and replace with commas
                    line = line.replace(’  ’,’’)
                    line = line.replace(’ ’,’,’)
                    line = line.replace(’ ’,’,’)
                    line = line.replace(’,,energy,’,’energy’)
                    line = line.replace(’,,total,,’,’total,’)
                    # write the data to analysis file in CSV format
                    o.write(line)

# Close and reopen the file to flush the buffer
    o.close()
    o = open(outFile,"r")

# Normalize data from Stage 2 runs

if aType == "XRF-S2" or aType == "KED-S2":
    # import the source data from the preprocessed output file
    spectrumIn = np.genfromtxt(outFile, delimiter=",", skip_header=3,
        skip_footer=1,
            usecols=(0,1))
    #print spectrumIn
    #print "spectrumIn=",len(spectrumIn)

    s2Bins = np.genfromtxt(outFile, delimiter=",", skip_header=3,
        skip_footer=1,
            usecols=(0))
```

```
s2Tally = np.genfromtxt(outFile, delimiter=",", skip_header=3,
    skip_footer=1,
        usecols=(1))
s2Error = np.genfromtxt(outFile, delimiter=",", skip_header=3,
    skip_footer=1,
        usecols=(2))

# Convert Stage 2 bin energies to keV
# The average offset for the Ka1 and Ka2 peaks is 0.1035 but 0.07
    should
# make the Kb's align more closely.
for i in range(0,len(s2Bins)):
    s2Bins[i] = s2Bins[i]*1000-0.07

# Initialize normalized tally matrix
s2NormTal = np.zeros((len(s2Tally),1))

s2CdPeak = "n"
# Search the S2 tally for the 88 keV peak
for i in range(0,len(s2Tally)):
    if s2CdPeak == "n":
        if s2Bins[i] > 88.03: #keV
            s2CdPeak = s2Tally[i]
            #s2CdPeak = s2Tally[i] - ((s2Tally[1150]+s2Tally[1171])
                /2)
            CdPeakLoc = i

# Normalize the Stage 2 tally
for i in range(0,len(s2Tally)):
    # Normalize to the Cd-109 peak
```

```
        s2NormTal[i] = s2Tally[i]/s2CdPeak
        #s2NormTal[i] = s2Tally[i]/sum(s2Tally)


# K-edge continuum comparison

if aType == "KED-S2":

    print "\n-=- K-EDGE CONTINUUM ANALYSIS -=-"

    # Set measured lower and upper continuum boundaries
    k_lc_l_model = 1444
    k_lc_u_model = 1517
    k_uc_l_model = 1540
    k_uc_u_model = 1612

    # Measured
    print "\n-=- MODELED CONTINUUM BOUNDS -=-"
    print "Modeled lower K-edge continuum lower bound at channel: ",
        k_lc_l_model,"(",s2Bins[k_lc_l_model]," keV )"
    print "Modeled lower K-edge continuum upper bound at channel: ",
        k_lc_u_model,"(",s2Bins[k_lc_u_model]," keV )"
    print "Modeled upper K-edge continuum lower bound at channel: ",
        k_uc_l_model,"(",s2Bins[k_uc_l_model]," keV )"
    print "Modeled upper K-edge continuum upper bound at channel: ",
        k_uc_u_model,"(",s2Bins[k_uc_u_model]," keV )"

    # Determine continuum area for modeled data
    k_lc_sum_model = np.sum(s2NormTal[k_lc_l_model:k_lc_u_model])
    k_uc_sum_model = np.sum(s2NormTal[k_uc_l_model:k_uc_u_model])
```

```
    k_ratio_model = k_lc_sum_model/k_uc_sum_model

    k_ratio_model_unc = np.average(np.average(s2Error[k_lc_l_model:
        k_lc_u_model])+np.average(s2Error[k_uc_l_model:k_uc_u_model]))

    # Print the results
    print "\n-=- K-EDGE ANALYSIS RESULTS -=-"
    print "Modeled K-edge continuum ratio: ", k_ratio_model
    print "Modeled K-edge continuum uncertainty: ",k_ratio_model_unc

    print "\nK-edge Excel data format"
    print k_ratio_model,k_ratio_model_unc


# Uranium XRF Peak area comparison

if aType == "XRF-S2":

    print "\n-=- URANIUM K ALPHA PEAK AREA ANALYSIS -=-"

    ## Set peak boundaries
    ## U K alpha peaks
    ka2_l = 93.9
    ka2_u = 95.36
    ka1_l = 97.6
    ka1_u = 99.1


    # Set tally channel variables to a string
```

```
ka2_l_chan_t = "n"
ka2_u_chan_t = "n"
ka1_l_chan_t = "n"
ka1_u_chan_t = "n"

# Tally
for i in range(0,len(s2Bins)-1):
    if (s2Bins[i] > ka2_l and ka2_l_chan_t == "n"):
        ka2_l_chan_t = i
        print "\nTally Ka2 lower bound at channel: ",ka2_l_chan_t
            ,"(",s2Bins[i]," keV )"
        #print s2NormTal[i]
    elif (s2Bins[i] > ka2_u and ka2_u_chan_t == "n"):
        ka2_u_chan_t = i
        print "Tally Ka2 upper bound at channel: ",ka2_u_chan_t,"(",
            s2Bins[i]," keV )"
        #print s2NormTal[i]
    elif (s2Bins[i] > ka1_l and ka1_l_chan_t == "n"):
        ka1_l_chan_t = i
        print "Tally Ka1 lower bound at channel: ",ka1_l_chan_t,"(",
            s2Bins[i]," keV )"
        #print s2NormTal[i]
    elif (s2Bins[i] > ka1_u and ka1_u_chan_t == "n"):
        ka1_u_chan_t = i
        print "Tally Ka1 upper bound at channel: ",ka1_u_chan_t,"(",
            s2Bins[i]," keV )"
        #print s2NormTal[i]

# Measure the peak areas
ka2_sum_tal = np.sum((s2NormTal[ka2_l_chan_t:ka2_u_chan_t]))
```

```
# Measure the peak areas
ka1_sum_tal = np.sum((s2NormTal[ka1_l_chan_t:ka1_u_chan_t]))

# Ka2 calculate the continuum
ka2_cont_t = ((s2NormTal[ka2_l_chan_t]+s2NormTal[ka2_u_chan_t])/2) *
     (ka2_u_chan_t-ka2_l_chan_t)
# Ka1 calculate the continuum
ka1_cont_t = ((s2NormTal[ka1_l_chan_t]+s2NormTal[ka1_u_chan_t])/2) *
     (ka1_u_chan_t-ka1_l_chan_t)

# Subtract the continuums from the peak areas
ka2_peak_t = ka2_sum_tal - ka2_cont_t
ka1_peak_t = ka1_sum_tal - ka1_cont_t

# U K beta peaks fit as doublets
kb13_l = 109.6
kb13_u = 111.8
kb24_l = 113.6
kb24_u = 116.1

# Set measured channel variables to a string
kb13_l_chan_m = "n"
kb13_u_chan_m = "n"
kb24_l_chan_m = "n"
kb24_u_chan_m = "n"

print "\n-=- URANIUM K BETA PEAK AREA ANALYSIS -=-"

# Set tally channel variables to a string
```

```
kb13_l_chan_t = "n"
kb13_u_chan_t = "n"
kb24_l_chan_t = "n"
kb24_u_chan_t = "n"

# Tally
for i in range(0,len(s2Bins)-1):
    if (s2Bins[i] > kb13_l and kb13_l_chan_t == "n"):
        kb13_l_chan_t = i
        print "\nTally Kb13 lower bound at channel: ",kb13_l_chan_t
            ,"(",s2Bins[i]," keV )"
        #print s2NormTal[i]
    elif (s2Bins[i] > kb13_u and kb13_u_chan_t == "n"):
        kb13_u_chan_t = i
        print "Tally Kb13 upper bound at channel: ",kb13_u_chan_t
            ,"(",s2Bins[i]," keV )"
        #print s2NormTal[i]
    elif (s2Bins[i] > kb24_l and kb24_l_chan_t == "n"):
        kb24_l_chan_t = i
        print "Tally Kb24 lower bound at channel: ",kb24_l_chan_t
            ,"(",s2Bins[i]," keV )"
        #print s2NormTal[i]
    elif (s2Bins[i] > kb24_u and kb24_u_chan_t == "n"):
        kb24_u_chan_t = i
        print "Tally Kb24 upper bound at channel: ",kb24_u_chan_t
            ,"(",s2Bins[i]," keV )"
        #print s2NormTal[i]

# Measure the peak areas
kb13_sum_tal = np.sum((s2NormTal[kb13_l_chan_t:kb13_u_chan_t]))
```

```
# Measure the peak areas
kb24_sum_tal = np.sum((s2NormTal[kb24_l_chan_t:kb24_u_chan_t]))

# kb13 calculate the continuum
kb13_cont_t = ((s2NormTal[kb13_l_chan_t]+s2NormTal[kb13_u_chan_t])
    /2) * (kb13_u_chan_t-kb13_l_chan_t)
# kb24 calculate the continuum
kb24_cont_t = ((s2NormTal[kb24_l_chan_t]+s2NormTal[kb24_u_chan_t])
    /2) * (kb24_u_chan_t-kb24_l_chan_t)

# Subtract the continuums from the peak areas
kb13_peak_t = kb13_sum_tal - kb13_cont_t
kb24_peak_t = kb24_sum_tal - kb24_cont_t



# Calculate the peak uncertainties
# Ka2
#ka2_peak_m_unc = np.sqrt(ka2_peak_m)
#ka2_peak_t_unc = np.sqrt(ka2_peak_t)
# Ka1
#ka1_peak_m_unc = np.sqrt(ka1_peak_m)
#ka1_peak_t_unc = np.sqrt(ka1_peak_t)
# Kb13
#kb13_peak_m_unc = np.sqrt(kb13_peak_m)
#kb13_peak_t_unc = np.sqrt(kb13_peak_t)
# Kb24
#kb24_peak_m_unc = np.sqrt(kb24_peak_m)
#kb24_peak_t_unc = np.sqrt(kb24_peak_t)
```

```
# New tally error calculation
ka2_peak_t_unc = np.average(s2Error[ka2_l_chan_t:ka2_u_chan_t])
ka1_peak_t_unc = np.average(s2Error[ka1_l_chan_t:ka1_u_chan_t])
kb13_peak_t_unc = np.average(s2Error[kb13_l_chan_t:kb13_u_chan_t])
kb24_peak_t_unc = np.average(s2Error[kb24_l_chan_t:kb24_u_chan_t])

print "\n-=- URANIUM PEAK AREAS -=-"
print "Ka2 tally: ",ka2_peak_t
print "Ka2 tally uncertainty: ",ka2_peak_t_unc

print "Ka1 tally: ",ka1_peak_t
print "Ka1 tally uncertainty: ",ka1_peak_t_unc

print "Kb13 tally: ",kb13_peak_t
print "Kb13 tally uncertainty: ",kb13_peak_t_unc

print "Kb24 tally: ",kb24_peak_t
print "Kb24 tally uncertainty: ",kb24_peak_t_unc

print "\n"

# Excel data format
print "\nUranium Excel data format\n"
#print ka1_peak_m+","+ka1_peak_m_unc+","+ka2_peak_m+","+
    ka2_peak_m_unc+","+kb13_peak_m+","+kb13_peak_m_unc+","+
    kb24_peak_m+","+kb24_peak_m_unc+","+ka1_peak_t+","+ka1_peak_t_unc
    +","+ka2_peak_t+","+ka2_peak_t_unc+","+kb13_peak_t+","+
    kb13_peak_t_unc+","+kb24_peak_t+","+kb24_peak_t_unc
```

```
    print ka1_peak_t,ka1_peak_t_unc,ka2_peak_t,ka2_peak_t_unc,
        kb13_peak_t,kb13_peak_t_unc,kb24_peak_t,kb24_peak_t_unc


# Plutionum XRF peaks

if aType == "XRF-S2" and sType == "U-Pu":

    print "\n-=- PLUTIONIUM K ALPHA PEAK AREA ANALYSIS -=-"

    # Set peak boundaries
    # Pu K alpha peaks
    ka2_l = 98.9
    ka2_u = 100.1
    ka1_l = 97.8
    ka1_u = 99.0

    # Set measured channel variables to a string
    ka2_l_chan_m = "n"
    ka2_u_chan_m = "n"
    ka1_l_chan_m = "n"
    ka1_u_chan_m = "n"


    # Set tally channel variables to a string
    ka2_l_chan_t = "n"
    ka2_u_chan_t = "n"
    ka1_l_chan_t = "n"
    ka1_u_chan_t = "n"
```

```
# Tally
for i in range(0,len(s2Bins)-1):
    if (s2Bins[i] > ka2_l and ka2_l_chan_t == "n"):
        ka2_l_chan_t = i
        print "\nTally Ka2 lower bound at channel: ",ka2_l_chan_t
            ,"(",s2Bins[i]," keV )"
        #print s2NormTal[i]
    elif (s2Bins[i] > ka2_u and ka2_u_chan_t == "n"):
        ka2_u_chan_t = i
        print "Tally Ka2 upper bound at channel: ",ka2_u_chan_t,"(",
            s2Bins[i]," keV )"
        #print s2NormTal[i]
    elif (s2Bins[i] > ka1_l and ka1_l_chan_t == "n"):
        ka1_l_chan_t = i
        print "Tally Ka1 lower bound at channel: ",ka1_l_chan_t,"(",
            s2Bins[i]," keV )"
        #print s2NormTal[i]
    elif (s2Bins[i] > ka1_u and ka1_u_chan_t == "n"):
        ka1_u_chan_t = i
        print "Tally Ka1 upper bound at channel: ",ka1_u_chan_t,"(",
            s2Bins[i]," keV )"
        #print s2NormTal[i]

# Measure the peak areas
ka2_sum_tal = np.sum((s2NormTal[ka2_l_chan_t:ka2_u_chan_t]))

# Measure the peak areas
ka1_sum_tal = np.sum((s2NormTal[ka1_l_chan_t:ka1_u_chan_t]))

# Ka2 calculate the continuum
```

```
ka2_cont_t = ((s2NormTal[ka2_l_chan_t]+s2NormTal[ka2_u_chan_t])/2) *
     (ka2_u_chan_t-ka2_l_chan_t)
# Ka1 calculate the continuum
ka1_cont_t = ((s2NormTal[ka1_l_chan_t]+s2NormTal[ka1_u_chan_t])/2) *
     (ka1_u_chan_t-ka1_l_chan_t)

# Subtract the continuums from the peak areas
ka2_peak_t = ka2_sum_tal - ka2_cont_t
ka1_peak_t = ka1_sum_tal - ka1_cont_t

if ka2_peak_t < 0:
    ka2_cont_t = 0
    ka2_peak_t = 0 #ka2_sum_tal - ka2_cont_t
if ka1_peak_t < 0:
    ka1_cont_t = 0
    ka1_peak_t = 0 #ka1_sum_tal - ka1_cont_t

# Pu K beta peaks fit as doublets
kb13_l = 110.0
kb13_u = 111.6
kb24_l = 114.0
kb24_u = 116.1

# Set measured channel variables to a string
kb13_l_chan_m = "n"
kb13_u_chan_m = "n"
kb24_l_chan_m = "n"
kb24_u_chan_m = "n"

print "\n-=- PLUTONIUM K BETA PEAK AREA ANALYSIS -=-"
```

```
# Set tally channel variables to a string
kb13_l_chan_t = "n"
kb13_u_chan_t = "n"
kb24_l_chan_t = "n"
kb24_u_chan_t = "n"

# Tally
for i in range(0,len(s2Bins)-1):
    if (s2Bins[i] > kb13_l and kb13_l_chan_t == "n"):
        kb13_l_chan_t = i
        print "\nTally Kb13 lower bound at channel: ",kb13_l_chan_t
            ,"(",s2Bins[i]," keV )"
        #print s2NormTal[i]
    elif (s2Bins[i] > kb13_u and kb13_u_chan_t == "n"):
        kb13_u_chan_t = i
        print "Tally Kb13 upper bound at channel: ",kb13_u_chan_t
            ,"(",s2Bins[i]," keV )"
        #print s2NormTal[i]
    elif (s2Bins[i] > kb24_l and kb24_l_chan_t == "n"):
        kb24_l_chan_t = i
        print "Tally Kb24 lower bound at channel: ",kb24_l_chan_t
            ,"(",s2Bins[i]," keV )"
        #print s2NormTal[i]
    elif (s2Bins[i] > kb24_u and kb24_u_chan_t == "n"):
        kb24_u_chan_t = i
        print "Tally Kb24 upper bound at channel: ",kb24_u_chan_t
            ,"(",s2Bins[i]," keV )"
        #print s2NormTal[i]
```

```
# Measure the peak areas
kb13_sum_tal = np.sum((s2NormTal[kb13_l_chan_t:kb13_u_chan_t]))

# Measure the peak areas
kb24_sum_tal = np.sum((s2NormTal[kb24_l_chan_t:kb24_u_chan_t]))

# kb13 calculate the continuum
kb13_cont_t = ((s2NormTal[kb13_l_chan_t]+s2NormTal[kb13_u_chan_t])
    /2) * (kb13_u_chan_t-kb13_l_chan_t)
# kb24 calculate the continuum
kb24_cont_t = ((s2NormTal[kb24_l_chan_t]+s2NormTal[kb24_u_chan_t])
    /2) * (kb24_u_chan_t-kb24_l_chan_t)



# Subtract the continuums from the peak areas
kb13_peak_t = kb13_sum_tal - kb13_cont_t
kb24_peak_t = kb24_sum_tal - kb24_cont_t



if kb13_peak_t < 0:
    kb13_cont_t = 0
    kb13_peak_t = 0 #ka2_sum_tal - ka2_cont_t
if ka1_peak_t < 0:
    kb24_cont_t = 0
    kb24_peak_t = 0 #ka1_sum_tal - ka1_cont_t

# Calculate the peak uncertainties
# Ka2
#ka2_peak_m_unc = np.sqrt(ka2_peak_m)
```

```
#ka2_peak_t_unc = np.sqrt(ka2_peak_t)
# Ka1
#ka1_peak_m_unc = np.sqrt(ka1_peak_m)
#ka1_peak_t_unc = np.sqrt(ka1_peak_t)
# Kb13
#kb13_peak_m_unc = np.sqrt(kb13_peak_m)
#kb13_peak_t_unc = np.sqrt(kb13_peak_t)
# Kb24
#kb24_peak_m_unc = np.sqrt(kb24_peak_m)
#kb24_peak_t_unc = np.sqrt(kb24_peak_t)



# New tally error calculation
ka2_peak_t_unc = np.average(s2Error[ka2_l_chan_t:ka2_u_chan_t])
ka1_peak_t_unc = np.average(s2Error[ka1_l_chan_t:ka1_u_chan_t])
kb13_peak_t_unc = np.average(s2Error[kb13_l_chan_t:kb13_u_chan_t])
kb24_peak_t_unc = np.average(s2Error[kb24_l_chan_t:kb24_u_chan_t])

#ka1_diff = (ka1_peak_m-ka1_peak_t)/ka1_peak_m
#ka2_diff = (ka2_peak_m-ka2_peak_t)/ka2_peak_m
#kb13_diff = (kb13_peak_m-kb13_peak_t)/kb13_peak_m
#kb24_diff = (kb24_peak_m-kb24_peak_t)/kb24_peak_m

print "\n-=- PLUTONIUM PEAK AREAS -=-"

print "Ka2 tally: ",ka2_peak_t
print "Ka2 tally uncertainty: ",ka2_peak_t_unc

print "Ka1 tally: ",ka1_peak_t
print "Ka1 tally uncertainty: ",ka1_peak_t_unc
```

```
    print "Kb13 tally: ",kb13_peak_t
    print "Kb13 tally uncertainty: ",kb13_peak_t_unc

    print "Kb24 tally: ",kb24_peak_t
    print "Kb24 tally uncertainty: ",kb24_peak_t_unc

    #print "\nKa1 error: ",ka1_diff*100," %"
    #print "Ka2 error: ",ka2_diff*100," %"
    #print "Kb13 error: ",kb13_diff*100," %"
    #print "Kb24 error: ",kb24_diff*100," %"
    #print "\n"

    # Excel data format
    print "\nPlutonium Excel data format\n"
    #print ka1_peak_m+","+ka1_peak_m_unc+","+ka2_peak_m+","+
        ka2_peak_m_unc+","+kb13_peak_m+","+kb13_peak_m_unc+","+
        kb24_peak_m+","+kb24_peak_m_unc+","+ka1_peak_t+","+ka1_peak_t_unc
        +","+ka2_peak_t+","+ka2_peak_t_unc+","+kb13_peak_t+","+
        kb13_peak_t_unc+","+kb24_peak_t+","+kb24_peak_t_unc
    print ka1_peak_t,ka1_peak_t_unc,ka2_peak_t,ka2_peak_t_unc,
        kb13_peak_t,kb13_peak_t_unc,kb24_peak_t,kb24_peak_t_unc



# Plot the results

chan = range(0,2048)
```

```
# correct the residuals bins by 4 keV
if aType == "KED-S2":
    s2ResidBins = s2Bins[:] - 4
elif aType == "XRF-S2":
    s2ResidBins = s2Bins[:] + 3

print "\n-=- PLOTTING -=-"
# If plotResults is "y" then call matplotlib
if plotResults == "y":

    # close any previous figures
    plt.close("all")

    # Prompt for plot title
    plotTitle = "" #raw_input("Enter string for plot title: ")

    # Set the figure dimensions
    plt.figure(figsize=(8,4))

    # Plot the tally results
    plt.subplot(1,1,1)
    plt.ylabel("Relative Intensity")
    plt.xlim((0,155))
    plt.ylim((1E-4,1E2))
    plt.semilogy(s2Bins,s2NormTal,color="black")

    # Set the title
    plt.title(plotTitle)
```

```
#plt.xlim((0,155))
plt.ylabel("Relative Intensity", fontsize=10)
plt.xlabel("Energy (keV)", fontsize=10)

# Set the legend location
plt.legend(loc=1,prop={'size':6})
plt.tick_params(labelsize=12)
plt.tick_params(which='both', width=1, labelsize=10)
plt.tick_params(which='major', length=8)
plt.grid(b=True,which="major")
if aType == "XRF-S2":
    if sType == "U":
        # U case
        plt.axvspan(93,100,facecolor='0.5',alpha=0.25)
        plt.axvspan(108,117,facecolor='0.5',alpha=0.25)
    elif sType == "U-Pu":
        # U-Pu case
        plt.axvspan(93.6,95.5,facecolor='0.5',alpha=0.25)
        plt.axvspan(97.6,99.09,facecolor='0.5',alpha=0.25)
        plt.axvspan(108.96,112.1,facecolor='0.5',alpha=0.25)
        plt.axvspan(113.6,115.7,facecolor='0.5',alpha=0.25)
        plt.axvspan(99.1,100,facecolor='g',alpha=0.25)
        plt.axvspan(103,104.5,facecolor='g',alpha=0.25)
        plt.axvspan(115.75,118,facecolor='g',alpha=0.25)
        plt.axvspan(119.5,121.5,facecolor='g',alpha=0.25)
        plt.axvspan(96.5,97.4,facecolor='b',alpha=0.25)
        plt.axvspan(100.3,101.6,facecolor='b',alpha=0.25)
        plt.axvspan(112.2,113.5,facecolor='b',alpha=0.25)
elif aType == "KED-S2":
    plt.axvspan(114,117,facecolor='0.5',alpha=0.25)
```

```
        plt.axvspan(120.6,123,facecolor='g',alpha=0.25)



    # Ask user to save the file
    #saveFile = "y" #raw_input("Do you want to save full figure as a PDF
        ? (y/n): ")
    if saveFile == "y":
        #figName = raw_input("Enter file name: ")
        plt.savefig(figName, dpi=1000, format='pdf', orientation='
            landscape',
            bbox_inches='tight')
        plt.savefig(figName2, dpi=1000, format='png', orientation='
            landscape',
            bbox_inches='tight')
        print "INFO: Full figures saved"
    elif saveFile =="n":
        print "INFO: Full figures not saved"

#######
# Plot detailed version
# Set the figure dimensions
    plt.figure(figsize=(8,4))

    # Plot the tally results
    plt.subplot(1,1,1)
    plt.ylabel("Relative Intensity")
    plt.xlabel("Energy (keV)", fontsize=10)

    plt.ylim((1E-4,1E2))
```

```
plt.semilogy(s2Bins,s2NormTal,color="black")
#plt.errorbar(s2Bins,s2Tally,yerr=corrError*s2Tally,color="red",
#    label="MCNP",errorevery=10)
## Set the title
plt.title(plotTitle)
plt.xlim((92,122))
plt.ylabel("Relative Intensity", fontsize=10)
# Set the legend location
plt.legend(loc=1,prop={'size':6})
plt.tick_params(labelsize=12)
plt.tick_params(which='both', width=1, labelsize=10)
plt.tick_params(which='major', length=8)
plt.grid(b=True,which="major")
if aType == "XRF-S2":
    if sType == "U":
        # U case
        plt.axvspan(93,100,facecolor='0.5',alpha=0.25)
        plt.axvspan(108,117,facecolor='0.5',alpha=0.25)
    elif sType == "U-Pu":
        # U-Pu case
        plt.axvspan(93.6,95.5,facecolor='0.5',alpha=0.25)
        plt.axvspan(97.6,99.09,facecolor='0.5',alpha=0.25)
        plt.axvspan(108.96,112.1,facecolor='0.5',alpha=0.25)
        plt.axvspan(113.6,115.7,facecolor='0.5',alpha=0.25)
        plt.axvspan(99.1,100,facecolor='g',alpha=0.25)
        plt.axvspan(103,104.5,facecolor='g',alpha=0.25)
        plt.axvspan(115.75,118,facecolor='g',alpha=0.25)
        plt.axvspan(119.5,121.5,facecolor='g',alpha=0.25)
        plt.axvspan(96.5,97.4,facecolor='b',alpha=0.25)
        plt.axvspan(100.3,101.6,facecolor='b',alpha=0.25)
```

```
            plt.axvspan(112.2,113.5,facecolor='b',alpha=0.25)
    elif aType == "KED-S2":
        plt.axvspan(114,117,facecolor='0.5',alpha=0.25)
        plt.axvspan(120.6,123,facecolor='g',alpha=0.25)



    # Ask user to save the file
    #saveFile = "y" #raw_input("Do you want to save zoomed figure as a
        PDF? (y/n): ")
    if saveFile == "y":
        #figNameROI = raw_input("Enter file name: ")
        plt.savefig(figNameROI, dpi=1000, format='pdf', orientation='
            landscape',
            bbox_inches='tight')
        plt.savefig(figNameROI2, dpi=1000, format='png', orientation='
            landscape',
            bbox_inches='tight')
        print "INFO: Detailed figures saved"
    elif saveFile =="n":
        print "INFO: Detailed figures not saved"

## Print message if not plotting results
elif plotResults == "n":
    print "INFO: Results not plotted"

plt.show()



# close all open files
```

```
f.close()
o.close()

if stage == "s2":
    r.close()
#s.close()
```

### C.1.3 Surrogate Samples

```
# -*- coding: utf-8 -*-
# HPAT: HKED Python Analysis Tool Pyro
# Version 2.0
#    Matthew T. Cook
# 12 March 2014
# Department of Nuclear Engineering
# University of Tennessee, Knoxville

# import modules
import os as os
import numpy as np
import matplotlib.pyplot as plt
import warnings


# Warning control
# Set script to ignore runtime warnings from modelResid divide by zero
warnings.simplefilter("ignore")


# Clear the screen
os.system('clear')
```

```
# Identify script
print " ---------------------------------"
print "| HKED Python Analysis Tool v. 2.0|"
print " ---------------------------------"

# Run in automatic
auto = "n" # "y" or "n"

# Save plot files?
saveFile = "y" # "y" or "n"

plotResults = "y"

# Concentration
conc = "250"
run  = "ked"  # "xrf" or "ked"
#sampType = "u" # "u" or "upu"
stage = "s2"  # "s1" or "s2"
# Hardcode sample type
sType = "U-Th"  # Options: "U" or "U-Th"
ratio = "_1_1"
cat = "surrogates"

if stage == "s1":
    outdir = "s1_cases"
elif stage == "s2":
    outdir = "s2_cases"

# Hardcode file name in for now
```

```
specFile = "data/"+run+"_"+conc+"gL_"+sType+".txt"
#specFile = "data/old/300gL_ked_u.txt"

fileName = outdir+"/"+run+"/"+cat+"/hked_v11_"+run+"_"+stage+"_"+conc+"
    gL_"+sType+ratio+".inp.o"




name = conc+"gL_"+run+"_"+sType+"_"+ratio

# Hardcode figure file names
figName = name + "_full.pdf"
figNameROI = name + "_detail.pdf"
figName2 = name + "_full.png"
figNameROI2 = name + "_detail.png"



# Open and create files needed by this script
# Open the input file for reading
f = open(fileName, "r")

# rename the input file and open it as a post processed output
base = os.path.splitext(fileName)[0]
outFile = base + ".opp"
sourceFile = base + ".src"
if stage == "s2":
    resultsFile = "RawResults_KED_surrogates_ratio.txt"
    r = open(resultsFile, "a")
    r.write(name)
```

```
    r.write(",")

# create and open the output files for writing
o = open(outFile, "w")
#s = open(sourceFile, "wb")


# Determine the run type and search terms

# Tell the user what's going on
print "\nProcessing MCNP output:", fileName
# Search the file for XRF-RUN or KED-RUN tags
with f as search:
    for line in search:
        # Remove '\n' at end of line
        line = line.rstrip()
        if "xrf-s1" in line:
            aType = "XRF-S1"
            print "Run Type:", aType
            break
        elif "xrf-s2" in line:
            aType = "XRF-S2"
            print "Run Type:", aType
            break
        elif "ked-s1" in line:
            aType = "KED-S1"
            print "Run Type:", aType
            break
        elif "ked-s2" in line:
            aType = "KED-S2"
```

```
            print "Run Type:", aType
            break

if aType == "XRF-S2" or aType == "KED-S2":
    # Ask the user what to do
    #plotResults = "n" #raw_input("Plot the results? (y/n): ")
    #plotResults = "n"

    # Ask user if a measured spectrum is to be analyzed
    importSpectrum = "y" #raw_input("Analyze a measured spectrum? (y/n):
        ")
if aType == "XRF-S1" or aType == "KED-S1":
    importSpectrum = "n"
    plotResults = "n"


# Tell the user where the extracted spectrum is
print "MCNP tally extracted to:",outFile

# Set the tally surface/volume search string
if aType == "XRF-S1":
    searchPhrase = " surface 608.2"
    print "Searched MCNP output for:", searchPhrase
elif aType == "XRF-S2":
    searchPhrase = " cell 75"
    print "Searched MCNP output for:", searchPhrase
elif aType == "KED-S1":
    searchPhrase = " surface 311.2"
    print "Searched MCNP output for:", searchPhrase
elif aType == "KED-S2":
```

```
    searchPhrase = " cell 44"
    print "Searched MCNP output for:", searchPhrase


# Create an output file and extract spectrum from MCNP output

if aType == "XRF-S1" or aType == "KED-S1":
    # Identify Stage 1 analysis
    print "Extracting Stage 1 F2 tally..."
    # Search for the specified string in the input file
    with open(fileName) as search:
        for line in search:
            # Remove '\n' at end of line
            line = line.rstrip()
            if searchPhrase in line:
                #for x in range (0,2053):
                #for x in range (0,602):
                #for x in range (0,2051):
                for x in range (0,8194):
                        line = search.next()
                        # Remove the spaces between tally values
                        # and replace with commas
                        line = line.replace('  ','')
                        line = line.replace(' ',',')
                        line = line.replace(' ',',')
                        line = line.replace(',,energy,','energy')
                        line = line.replace(',,total,,',',total,')
                        # Write the data to analysis file in CSV format
                        o.write(line)
    # Close and reopen output file to flush the buffer
```

```
    o.close()
    o = open(outFile,"r")


# Create source file from Stage 1 tally for Stage 2 source

# write the Stage 2 SDEF options to the new source file with a Cd-109
    check source
if aType == "XRF-S1":
    s = open(sourceFile, "wb")
    s2sdef = "SDEF VEC=-0.692 -0.721 0 DIR=1 POS=D1 ERG=FPOS=D2 PAR=2
        ARA=0.001 \n"
    s2sdef = s2sdef + "SI1 L -9.57 -9.97 0   -9.57 -9.97 0 \n"
    s2sdef = s2sdef + "SP1   1               1E-3 \n"
    s2sdef = s2sdef + "DS2 S 4               3 \n"
    s2sdef = s2sdef + "SI3 L 0.02199 0.022163 0.024912 0.02943 0.025455
        0.0880336 \n"
    s2sdef = s2sdef + "SP3 D 0.298  0.561   0.048   0.092  0.0231
        0.0370"
elif aType =="KED-S1":
    s = open(sourceFile, "wb")
    s2sdef = "SDEF VEC=1 0 0 DIR=1 POS=D1 ERG=FPOS=D2 PAR=2 ARA=0.001 \n
        "
    s2sdef = s2sdef + "SI1 L 19.8545 0 0   19.8545 1 0 \n"
    s2sdef = s2sdef + "SP1   1               2E-1 \n"
    s2sdef = s2sdef + "DS2 S 4               3 \n"
    s2sdef = s2sdef + "SI3 L 0.02199 0.022163 0.024912 0.02943 0.025455
        0.0880336 \n"
    s2sdef = s2sdef + "SP3 D 0.298  0.561   0.048   0.092  0.0231
        0.0370"
```

```
if aType == "XRF-S1" or aType == "KED-S1":
    print "Writing Stage 2 source file..."
    # write source file header
    sheader = "c Source file processed from output: "
    s.write(sheader + fileName + "\n")
    s.write(s2sdef + "\n")

    # import the source data from the preprocessed output file
    sourcein = np.genfromtxt(outFile, delimiter=",", skip_header=1,
        skip_footer=1,
            usecols=(0,1))
    mcnpEnergy = np.genfromtxt(outFile, delimiter=",", skip_header=1,
        skip_footer=1,
            usecols=(0))
    mcnpTally = np.genfromtxt(outFile, delimiter=",", skip_header=1,
        skip_footer=1,
            usecols=(1))
    mcnpError= np.genfromtxt(outFile, delimiter=",", skip_header=1,
        skip_footer=1,
            usecols=(2))

    # sum the source and copy the energies
    sumsource = sum(sourcein[:,1])
    sourceenergy = sourcein[:,0]

    # normalize the Stage 1 tally
    for x in range(0,len(sourcein)):
        sourcenorm = sourcein[:,1]/sumsource
```

```
# Polynomial fit parameters
#p1  =   5.347e-06
#p2  =   5.042e-06
#p3  =  -3.899e-05
#p4  =  -3.092e-05
#p5  =   8.373e-05
#p6  =   5.541e-05
#p7  =  -5.538e-05
#p8  =   -5.86e-05
#p9  =  -0.0002095
#p10 = -0.0002799
p1 = -0.2099
p2 =  0.07759
p3 = -0.0003616
p4 = -2.786E-5

# Correct the energy
for i in range(0,len(sourceenergy)):
    sourceenergy[i] = sourceenergy[i] - (p1*sourceenergy[i]**3 + p2*
        sourceenergy[i]**2 + p3*sourceenergy[i] + p4)

# write source energies
for x in range(0,len(sourcein)):
    if x == 0:
        #soutline1 = "SI1 L "
        # Use SI1 L for source with Cd-109
        soutline1 = "SI4 L "
        soutline2 = str(sourceenergy[x])
        s.write(soutline1)
        s.write(soutline2 + "\n")
```

```
        else:
            soutline = ("     " + str(sourceenergy[x]))
            s.write(soutline + "\n")

    # write the source intensities
    for x in range(0,len(sourcein)):
        if x == 0:
            #soutline1 = "SP1 D "
            # Use SP4 D for source with Cd-109
            soutline1 = "SP4 D "
            soutline2 = str(sourcenorm[x])
            s.write(soutline1)
            s.write(soutline2 + "\n")
        else:
            soutline = ("     " + str(sourcenorm[x]))
            s.write(soutline + "\n")

    # tell user where the source file was written
    print "Stage 1 tally written to Stage 2 source file:", sourceFile
    s.close()

    errorin = np.genfromtxt(outFile, delimiter=",", skip_header=1,
        skip_footer=1,
            usecols=(0,2))


# Import data from Stage 2 runs

if aType == "XRF-S2" or aType == "KED-S2":
    print "Extracting Stage 2 F8 tally..."
```

```
    with open(fileName) as search:
        for line in search:
            line = line.rstrip() # remove '\n' at end of line
            if searchPhrase in line:
                for x in range (0,2052):
                    line = search.next()
                    # remove the spaces between tally values
                    # and replace with commas
                    line = line.replace('  ','')
                    line = line.replace(' ',',')
                    line = line.replace(' ',',')
                    line = line.replace(',,energy,','energy')
                    line = line.replace(',,total,,',',total,')
                    # write the data to analysis file in CSV format
                    o.write(line)

# Close and reopen the file to flush the buffer
    o.close()
    o = open(outFile,"r")

# Normalize data from Stage 2 runs

if aType == "XRF-S2" or aType == "KED-S2":
    # import the source data from the preprocessed output file
    spectrumIn = np.genfromtxt(outFile, delimiter=",", skip_header=3,
        skip_footer=1,
            usecols=(0,1))
    #print spectrumIn
    #print "spectrumIn=",len(spectrumIn)
```

```
s2Bins = np.genfromtxt(outFile, delimiter=",", skip_header=3,
    skip_footer=1,
        usecols=(0))
s2Tally = np.genfromtxt(outFile, delimiter=",", skip_header=3,
    skip_footer=1,
        usecols=(1))
s2Error = np.genfromtxt(outFile, delimiter=",", skip_header=3,
    skip_footer=1,
        usecols=(2))

# Convert Stage 2 bin energies to keV
# The average offset for the Ka1 and Ka2 peaks is 0.1035 but 0.07
    should
# make the Kb's align more closely.
for i in range(0,len(s2Bins)):
    s2Bins[i] = s2Bins[i]*1000-0.07

# Initialize normalized tally matrix
s2NormTal = np.zeros((len(s2Tally),1))

s2CdPeak = "n"
# Search the S2 tally for the 88 keV peak
for i in range(0,len(s2Tally)):
    if s2CdPeak == "n":
        if s2Bins[i] > 88.03: #keV
            s2CdPeak = s2Tally[i]
            #s2CdPeak = s2Tally[i] - ((s2Tally[1150]+s2Tally[1171])
                /2)
            CdPeakLoc = i
```

```
    # Normalize the Stage 2 tally
    for i in range(0,len(s2Tally)):
        # Normalize to the Cd-109 peak
        s2NormTal[i] = s2Tally[i]/s2CdPeak
        #s2NormTal[i] = s2Tally[i]/sum(s2Tally)


# K-edge continuum comparison

if aType == "KED-S2":

    print "\n-=- K-EDGE CONTINUUM ANALYSIS -=-"

    # Set measured lower and upper continuum boundaries
    k_lc_l_model = 1444
    k_lc_u_model = 1517
    k_uc_l_model = 1540
    k_uc_u_model = 1612

    # Measured
    print "\n-=- MODELED CONTINUUM BOUNDS -=-"
    print "Modeled lower K-edge continuum lower bound at channel: ",
        k_lc_l_model,"(",s2Bins[k_lc_l_model]," keV )"
    print "Modeled lower K-edge continuum upper bound at channel: ",
        k_lc_u_model,"(",s2Bins[k_lc_u_model]," keV )"
    print "Modeled upper K-edge continuum lower bound at channel: ",
        k_uc_l_model,"(",s2Bins[k_uc_l_model]," keV )"
    print "Modeled upper K-edge continuum upper bound at channel: ",
        k_uc_u_model,"(",s2Bins[k_uc_u_model]," keV )"
```

```
    # Determine continuum area for modeled data
    k_lc_sum_model = np.sum(s2NormTal[k_lc_l_model:k_lc_u_model])
    k_uc_sum_model = np.sum(s2NormTal[k_uc_l_model:k_uc_u_model])

    k_ratio_model = k_lc_sum_model/k_uc_sum_model

    k_ratio_model_unc = np.average(np.average(s2Error[k_lc_l_model:
        k_lc_u_model])+np.average(s2Error[k_uc_l_model:k_uc_u_model]))

    # Print the results
    print "\n-=- K-EDGE ANALYSIS RESULTS -=-"
    print "Modeled K-edge continuum ratio: ", k_ratio_model
    print "Modeled K-edge continuum uncertainty: ",k_ratio_model_unc

    print "\nK-edge Excel data format"
    print k_ratio_model,k_ratio_model_unc
    rout = str(k_ratio_model)+","+str(k_ratio_model_unc)+"\n"
    r.write(rout)


# Uranium XRF Peak area comparison

if aType == "XRF-S2":

    print "\n-=- URANIUM K ALPHA PEAK AREA ANALYSIS -=-"

    ## Set peak boundaries
    ## U K alpha peaks
    ka2_l = 93.9
    ka2_u = 95.36
```

```
ka1_l = 97.6
ka1_u = 99.1


# Set tally channel variables to a string
ka2_l_chan_t = "n"
ka2_u_chan_t = "n"
ka1_l_chan_t = "n"
ka1_u_chan_t = "n"

# Tally
for i in range(0,len(s2Bins)-1):
    if (s2Bins[i] > ka2_l and ka2_l_chan_t == "n"):
        ka2_l_chan_t = i
        print "\nTally Ka2 lower bound at channel: ",ka2_l_chan_t
            ,"(",s2Bins[i]," keV )"
        #print s2NormTal[i]
    elif (s2Bins[i] > ka2_u and ka2_u_chan_t == "n"):
        ka2_u_chan_t = i
        print "Tally Ka2 upper bound at channel: ",ka2_u_chan_t,"(",
            s2Bins[i]," keV )"
        #print s2NormTal[i]
    elif (s2Bins[i] > ka1_l and ka1_l_chan_t == "n"):
        ka1_l_chan_t = i
        print "Tally Ka1 lower bound at channel: ",ka1_l_chan_t,"(",
            s2Bins[i]," keV )"
        #print s2NormTal[i]
    elif (s2Bins[i] > ka1_u and ka1_u_chan_t == "n"):
        ka1_u_chan_t = i
```

```
        print "Tally Ka1 upper bound at channel: ",ka1_u_chan_t,"(",
            s2Bins[i]," keV )"
        #print s2NormTal[i]

# Measure the peak areas
ka2_sum_tal = np.sum((s2NormTal[ka2_l_chan_t:ka2_u_chan_t]))

# Measure the peak areas
ka1_sum_tal = np.sum((s2NormTal[ka1_l_chan_t:ka1_u_chan_t]))

# Ka2 calculate the continuum
ka2_cont_t = ((s2NormTal[ka2_l_chan_t]+s2NormTal[ka2_u_chan_t])/2) *
     (ka2_u_chan_t-ka2_l_chan_t)
# Ka1 calculate the continuum
ka1_cont_t = ((s2NormTal[ka1_l_chan_t]+s2NormTal[ka1_u_chan_t])/2) *
     (ka1_u_chan_t-ka1_l_chan_t)

# Subtract the continuums from the peak areas
ka2_peak_t = ka2_sum_tal - ka2_cont_t
ka1_peak_t = ka1_sum_tal - ka1_cont_t

# U K beta peaks fit as doublets
kb13_l = 109.6
kb13_u = 111.8
kb24_l = 113.6
kb24_u = 116.1

# Set measured channel variables to a string
kb13_l_chan_m = "n"
kb13_u_chan_m = "n"
```

```
kb24_l_chan_m = "n"
kb24_u_chan_m = "n"

print "\n-=- URANIUM K BETA PEAK AREA ANALYSIS -=-"

# Set tally channel variables to a string
kb13_l_chan_t = "n"
kb13_u_chan_t = "n"
kb24_l_chan_t = "n"
kb24_u_chan_t = "n"

# Tally
for i in range(0,len(s2Bins)-1):
    if (s2Bins[i] > kb13_l and kb13_l_chan_t == "n"):
        kb13_l_chan_t = i
        print "\nTally Kb13 lower bound at channel: ",kb13_l_chan_t
            ,"(",s2Bins[i]," keV )"
        #print s2NormTal[i]
    elif (s2Bins[i] > kb13_u and kb13_u_chan_t == "n"):
        kb13_u_chan_t = i
        print "Tally Kb13 upper bound at channel: ",kb13_u_chan_t
            ,"(",s2Bins[i]," keV )"
        #print s2NormTal[i]
    elif (s2Bins[i] > kb24_l and kb24_l_chan_t == "n"):
        kb24_l_chan_t = i
        print "Tally Kb24 lower bound at channel: ",kb24_l_chan_t
            ,"(",s2Bins[i]," keV )"
        #print s2NormTal[i]
    elif (s2Bins[i] > kb24_u and kb24_u_chan_t == "n"):
        kb24_u_chan_t = i
```

```
        print "Tally Kb24 upper bound at channel: ",kb24_u_chan_t
            ,"(",s2Bins[i]," keV )"
        #print s2NormTal[i]

# Measure the peak areas
kb13_sum_tal = np.sum((s2NormTal[kb13_l_chan_t:kb13_u_chan_t]))

# Measure the peak areas
kb24_sum_tal = np.sum((s2NormTal[kb24_l_chan_t:kb24_u_chan_t]))

# kb13 calculate the continuum
kb13_cont_t = ((s2NormTal[kb13_l_chan_t]+s2NormTal[kb13_u_chan_t])
    /2) * (kb13_u_chan_t-kb13_l_chan_t)
# kb24 calculate the continuum
kb24_cont_t = ((s2NormTal[kb24_l_chan_t]+s2NormTal[kb24_u_chan_t])
    /2) * (kb24_u_chan_t-kb24_l_chan_t)

# Subtract the continuums from the peak areas
kb13_peak_t = kb13_sum_tal - kb13_cont_t
kb24_peak_t = kb24_sum_tal - kb24_cont_t



# Calculate the peak uncertainties
# Ka2
#ka2_peak_m_unc = np.sqrt(ka2_peak_m)
#ka2_peak_t_unc = np.sqrt(ka2_peak_t)
# Ka1
#ka1_peak_m_unc = np.sqrt(ka1_peak_m)
#ka1_peak_t_unc = np.sqrt(ka1_peak_t)
# Kb13
```

```
#kb13_peak_m_unc = np.sqrt(kb13_peak_m)
#kb13_peak_t_unc = np.sqrt(kb13_peak_t)
# Kb24
#kb24_peak_m_unc = np.sqrt(kb24_peak_m)
#kb24_peak_t_unc = np.sqrt(kb24_peak_t)



# New tally error calculation
ka2_peak_t_unc = np.average(s2Error[ka2_l_chan_t:ka2_u_chan_t])
ka1_peak_t_unc = np.average(s2Error[ka1_l_chan_t:ka1_u_chan_t])
kb13_peak_t_unc = np.average(s2Error[kb13_l_chan_t:kb13_u_chan_t])
kb24_peak_t_unc = np.average(s2Error[kb24_l_chan_t:kb24_u_chan_t])

print "\n-=- URANIUM PEAK AREAS -=-"
print "Ka2 tally: ",ka2_peak_t
print "Ka2 tally uncertainty: ",ka2_peak_t_unc

print "Ka1 tally: ",ka1_peak_t
print "Ka1 tally uncertainty: ",ka1_peak_t_unc

print "Kb13 tally: ",kb13_peak_t
print "Kb13 tally uncertainty: ",kb13_peak_t_unc

print "Kb24 tally: ",kb24_peak_t
print "Kb24 tally uncertainty: ",kb24_peak_t_unc

print "\n"

# Excel data format
print "\nUranium Excel data format\n"
```

```
    rout = str(ka1_peak_t)+","+str(ka1_peak_t_unc)+","+str(ka2_peak_t)
        +","+str(ka2_peak_t_unc)+","+str(kb13_peak_t)+","+str(
        kb13_peak_t_unc)+","+str(kb24_peak_t)+","+str(kb24_peak_t_unc)
        +","
    print ka1_peak_t,ka1_peak_t_unc,ka2_peak_t,ka2_peak_t_unc,
        kb13_peak_t,kb13_peak_t_unc,kb24_peak_t,kb24_peak_t_unc
    r.write(rout)


# Plutionum XRF peaks

if aType == "XRF-S2" and sType == "U-Th":

    print "\n-=- THORIUM K ALPHA PEAK AREA ANALYSIS -=-"

    # Set peak boundaries
    # Thorium K alpha peaks
    ka2_l = 89.2
    ka2_u = 90.4
    ka1_l = 92.8
    ka1_u = 93.8

    # Set measured channel variables to a string
    ka2_l_chan_m = "n"
    ka2_u_chan_m = "n"
    ka1_l_chan_m = "n"
    ka1_u_chan_m = "n"


    # Set tally channel variables to a string
```

```
ka2_l_chan_t = "n"
ka2_u_chan_t = "n"
ka1_l_chan_t = "n"
ka1_u_chan_t = "n"

# Tally
for i in range(0,len(s2Bins)-1):
    if (s2Bins[i] > ka2_l and ka2_l_chan_t == "n"):
        ka2_l_chan_t = i
        print "\nTally Ka2 lower bound at channel: ",ka2_l_chan_t
            ,"(",s2Bins[i]," keV )"
        #print s2NormTal[i]
    elif (s2Bins[i] > ka2_u and ka2_u_chan_t == "n"):
        ka2_u_chan_t = i
        print "Tally Ka2 upper bound at channel: ",ka2_u_chan_t,"(",
            s2Bins[i]," keV )"
        #print s2NormTal[i]
    elif (s2Bins[i] > ka1_l and ka1_l_chan_t == "n"):
        ka1_l_chan_t = i
        print "Tally Ka1 lower bound at channel: ",ka1_l_chan_t,"(",
            s2Bins[i]," keV )"
        #print s2NormTal[i]
    elif (s2Bins[i] > ka1_u and ka1_u_chan_t == "n"):
        ka1_u_chan_t = i
        print "Tally Ka1 upper bound at channel: ",ka1_u_chan_t,"(",
            s2Bins[i]," keV )"
        #print s2NormTal[i]

# Measure the peak areas
ka2_sum_tal = np.sum((s2NormTal[ka2_l_chan_t:ka2_u_chan_t]))
```

```
# Measure the peak areas
ka1_sum_tal = np.sum((s2NormTal[ka1_l_chan_t:ka1_u_chan_t]))

# Ka2 calculate the continuum
ka2_cont_t = ((s2NormTal[ka2_l_chan_t]+s2NormTal[ka2_u_chan_t])/2) *
     (ka2_u_chan_t-ka2_l_chan_t)
# Ka1 calculate the continuum
ka1_cont_t = ((s2NormTal[ka1_l_chan_t]+s2NormTal[ka1_u_chan_t])/2) *
     (ka1_u_chan_t-ka1_l_chan_t)

# Subtract the continuums from the peak areas
ka2_peak_t = ka2_sum_tal - ka2_cont_t
ka1_peak_t = ka1_sum_tal - ka1_cont_t

if ka2_peak_t < 0:
    ka2_cont_t = 0
    ka2_peak_t = 0 #ka2_sum_tal - ka2_cont_t
if ka1_peak_t < 0:
    ka1_cont_t = 0
    ka1_peak_t = 0 #ka1_sum_tal - ka1_cont_t

# Thorium K beta peaks fit as doublets
kb13_l = 103.98
kb13_u = 106.55
kb24_l = 114.0
kb24_u = 116.1

# Set measured channel variables to a string
kb13_l_chan_m = "n"
```

```
kb13_u_chan_m = "n"
kb24_l_chan_m = "n"
kb24_u_chan_m = "n"

print "\n-=- THORIUM K BETA PEAK AREA ANALYSIS -=-"



# Set tally channel variables to a string
kb13_l_chan_t = "n"
kb13_u_chan_t = "n"
kb24_l_chan_t = "n"
kb24_u_chan_t = "n"

# Tally
for i in range(0,len(s2Bins)-1):
    if (s2Bins[i] > kb13_l and kb13_l_chan_t == "n"):
        kb13_l_chan_t = i
        print "\nTally Kb13 lower bound at channel: ",kb13_l_chan_t
            ,"(",s2Bins[i]," keV )"
        #print s2NormTal[i]
    elif (s2Bins[i] > kb13_u and kb13_u_chan_t == "n"):
        kb13_u_chan_t = i
        print "Tally Kb13 upper bound at channel: ",kb13_u_chan_t
            ,"(",s2Bins[i]," keV )"
        #print s2NormTal[i]
    elif (s2Bins[i] > kb24_l and kb24_l_chan_t == "n"):
        kb24_l_chan_t = i
        print "Tally Kb24 lower bound at channel: ",kb24_l_chan_t
            ,"(",s2Bins[i]," keV )"
        #print s2NormTal[i]
```

```
    elif (s2Bins[i] > kb24_u and kb24_u_chan_t == "n"):
        kb24_u_chan_t = i
        print "Tally Kb24 upper bound at channel: ",kb24_u_chan_t
            ,"(",s2Bins[i]," keV )"
        #print s2NormTal[i]

# Measure the peak areas
kb13_sum_tal = np.sum((s2NormTal[kb13_l_chan_t:kb13_u_chan_t]))

# Measure the peak areas
kb24_sum_tal = np.sum((s2NormTal[kb24_l_chan_t:kb24_u_chan_t]))

# kb13 calculate the continuum
kb13_cont_t = ((s2NormTal[kb13_l_chan_t]+s2NormTal[kb13_u_chan_t])
    /2) * (kb13_u_chan_t-kb13_l_chan_t)
# kb24 calculate the continuum
kb24_cont_t = ((s2NormTal[kb24_l_chan_t]+s2NormTal[kb24_u_chan_t])
    /2) * (kb24_u_chan_t-kb24_l_chan_t)


# Subtract the continuums from the peak areas
kb13_peak_t = kb13_sum_tal - kb13_cont_t
kb24_peak_t = kb24_sum_tal - kb24_cont_t


if kb13_peak_t < 0:
    kb13_cont_t = 0
    kb13_peak_t = 0 #ka2_sum_tal - ka2_cont_t
if ka1_peak_t < 0:
    kb24_cont_t = 0
```

```
    kb24_peak_t = 0 #ka1_sum_tal - ka1_cont_t

# Calculate the peak uncertainties
# Ka2
#ka2_peak_m_unc = np.sqrt(ka2_peak_m)
#ka2_peak_t_unc = np.sqrt(ka2_peak_t)
# Ka1
#ka1_peak_m_unc = np.sqrt(ka1_peak_m)
#ka1_peak_t_unc = np.sqrt(ka1_peak_t)
# Kb13
#kb13_peak_m_unc = np.sqrt(kb13_peak_m)
#kb13_peak_t_unc = np.sqrt(kb13_peak_t)
# Kb24
#kb24_peak_m_unc = np.sqrt(kb24_peak_m)
#kb24_peak_t_unc = np.sqrt(kb24_peak_t)



# New tally error calculation
ka2_peak_t_unc = np.average(s2Error[ka2_l_chan_t:ka2_u_chan_t])
ka1_peak_t_unc = np.average(s2Error[ka1_l_chan_t:ka1_u_chan_t])
kb13_peak_t_unc = np.average(s2Error[kb13_l_chan_t:kb13_u_chan_t])
kb24_peak_t_unc = np.average(s2Error[kb24_l_chan_t:kb24_u_chan_t])

#ka1_diff = (ka1_peak_m-ka1_peak_t)/ka1_peak_m
#ka2_diff = (ka2_peak_m-ka2_peak_t)/ka2_peak_m
#kb13_diff = (kb13_peak_m-kb13_peak_t)/kb13_peak_m
#kb24_diff = (kb24_peak_m-kb24_peak_t)/kb24_peak_m

print "\n-=- THORIUM PEAK AREAS -=-"
```

```
print "Ka2 tally: ",ka2_peak_t
print "Ka2 tally uncertainty: ",ka2_peak_t_unc

print "Ka1 tally: ",ka1_peak_t
print "Ka1 tally uncertainty: ",ka1_peak_t_unc



print "Kb13 tally: ",kb13_peak_t
print "Kb13 tally uncertainty: ",kb13_peak_t_unc

print "Kb24 tally: ",kb24_peak_t
print "Kb24 tally uncertainty: ",kb24_peak_t_unc

#print "\nKa1 error: ",ka1_diff*100," %"
#print "Ka2 error: ",ka2_diff*100," %"
#print "Kb13 error: ",kb13_diff*100," %"
#print "Kb24 error: ",kb24_diff*100," %"
#print "\n"

# Excel data format
print "\nThorium Excel data format\n"
#rout = ka1_peak_t+","+ka1_peak_t_unc+","+ka2_peak_t+","+
    ka2_peak_t_unc+","+kb13_peak_t+","+kb13_peak_t_unc+","+
    kb24_peak_t+","+kb24_peak_t_unc
#print ka1_peak_t,ka1_peak_t_unc,ka2_peak_t,ka2_peak_t_unc,
    kb13_peak_t,kb13_peak_t_unc,kb24_peak_t,kb24_peak_t_unc
```

```
    rout = str(ka1_peak_t)+","+str(ka1_peak_t_unc)+","+str(ka2_peak_t)
        +","+str(ka2_peak_t_unc)+","+str(kb13_peak_t)+","+str(
        kb13_peak_t_unc)+","+str(kb24_peak_t)+","+str(kb24_peak_t_unc)+"\
        n"
    print ka1_peak_t,ka1_peak_t_unc,ka2_peak_t,ka2_peak_t_unc,
        kb13_peak_t,kb13_peak_t_unc,kb24_peak_t,kb24_peak_t_unc
    r.write(rout)


# Plot the results

chan = range(0,2048)

# correct the residuals bins by 4 keV
if aType == "KED-S2":
    s2ResidBins = s2Bins[:] - 4
elif aType == "XRF-S2":
    s2ResidBins = s2Bins[:] + 3

print "\n-=- PLOTTING -=-"
# If plotResults is "y" then call matplotlib
if plotResults == "y":

    # close any previous figures
    plt.close("all")

    # Prompt for plot title
    plotTitle = "" #raw_input("Enter string for plot title: ")

    # Set the figure dimensions
```

```
plt.figure(figsize=(8,4))

# Plot the tally results
plt.subplot(1,1,1)
plt.ylabel("Relative Intensity")
plt.xlim((0,155))
plt.ylim((1E-4,1E2))
plt.semilogy(s2Bins,s2NormTal,color="black")

# Set the title
plt.title(plotTitle)



#plt.xlim((0,155))
plt.ylabel("Relative Intensity", fontsize=10)
plt.xlabel("Energy (keV)", fontsize=10)
# Set the legend location
plt.legend(loc=1,prop={'size':6})
plt.tick_params(labelsize=12)
plt.tick_params(which='both', width=1, labelsize=10)
plt.tick_params(which='major', length=8)
plt.grid(b=True,which="major")
if aType == "XRF-S2":
    if sType == "U":
        # U case
        plt.axvspan(93,100,facecolor='0.5',alpha=0.25)
        plt.axvspan(108,117,facecolor='0.5',alpha=0.25)
    elif sType == "U-Th":
        # U-Th case
        plt.axvspan(93.85,99.53,facecolor='0.5',alpha=0.25)
```

```
            plt.axvspan(109.5,116.2,facecolor='0.5',alpha=0.25)
            plt.axvspan(89.2,93.8,facecolor='g',alpha=0.25)
            plt.axvspan(104.04,109.45,facecolor='g',alpha=0.25)
    elif aType == "KED-S2":
        plt.axvspan(107,111,facecolor='g',alpha=0.25)
        plt.axvspan(114,117,facecolor='0.5',alpha=0.25)

    # Ask user to save the file
    #saveFile = "y" #raw_input("Do you want to save full figure as a PDF
        ? (y/n): ")
    if saveFile == "y":
        #figName = raw_input("Enter file name: ")
        plt.savefig(figName, dpi=1000, format='pdf', orientation='
            landscape',
            bbox_inches='tight')
        plt.savefig(figName2, dpi=1000, format='png', orientation='
            landscape',
            bbox_inches='tight')
        print "INFO: Full figures saved"
    elif saveFile =="n":
        print "INFO: Full figures not saved"

#######
# Plot detailed version
# Set the figure dimensions
    plt.figure(figsize=(8,4))

    # Plot the tally results
    plt.subplot(1,1,1)
    plt.ylabel("Relative Intensity")
```

```
plt.ylim((1E-4,1E2))
plt.semilogy(s2Bins,s2NormTal,color="black")
#plt.errorbar(s2Bins,s2Tally,yerr=corrError*s2Tally,color="red",
#    label="MCNP",errorevery=10)
## Set the title
plt.title(plotTitle)
plt.xlim((88,117))
plt.ylabel("Relative Intensity", fontsize=10)
plt.xlabel("Energy (keV)", fontsize=10)
# Set the legend location
plt.legend(loc=1,prop={'size':6})
plt.tick_params(labelsize=12)
plt.tick_params(which='both', width=1, labelsize=10)
plt.tick_params(which='major', length=8)
plt.grid(b=True,which="major")
if aType == "XRF-S2":
    if sType == "U":
        # U case
        plt.axvspan(93,100,facecolor='0.5',alpha=0.25)
        plt.axvspan(108,117,facecolor='0.5',alpha=0.25)
    elif sType == "U-Th":
        # U-Th case
        plt.axvspan(93.85,99.53,facecolor='0.5',alpha=0.25)
        plt.axvspan(109.5,116.2,facecolor='0.5',alpha=0.25)
        plt.axvspan(89.2,93.8,facecolor='g',alpha=0.25)
        plt.axvspan(104.04,109.45,facecolor='g',alpha=0.25)
elif aType == "KED-S2":
    plt.axvspan(107,111,facecolor='g',alpha=0.25)
    plt.axvspan(114,117,facecolor='0.5',alpha=0.25)
```

```
    # Ask user to save the file
    #saveFile = "y" #raw_input("Do you want to save zoomed figure as a
        PDF? (y/n): ")
    if saveFile == "y":
        #figNameROI = raw_input("Enter file name: ")
        plt.savefig(figNameROI, dpi=1000, format='pdf', orientation='
            landscape',
            bbox_inches='tight')
        plt.savefig(figNameROI2, dpi=1000, format='png', orientation='
            landscape',
            bbox_inches='tight')
        print "INFO: Detailed figures saved"
    elif saveFile =="n":
        print "INFO: Detailed figures not saved"

## Print message if not plotting results
elif plotResults == "n":
    print "INFO: Results not plotted"


plt.show()



# close all open files
f.close()
o.close()

if stage == "s2":
    r.close()
#s.close()
```

### C.1.4 Complex Samples

```
# HPAT: HKED Python Analysis Tool Pyro
# Version 2.0
# Matthew T. Cook
# 12 March 2014
# Department of Nuclear Engineering
# University of Tennessee, Knoxville

# import modules
import os as os
import numpy as np
import matplotlib.pyplot as plt
import warnings



# Warning control
# Set script to ignore runtime warnings from modelResid divide by zero
warnings.simplefilter("ignore")



# Clear the screen
os.system('clear')

# Identify script
print " ---------------------------------"
print "| HKED Python Analysis Tool v. 2.0|"
print " ---------------------------------"

# Run in automatic
auto = "n" # "y" or "n"
```

```
# Save plot files?
saveFile = "y" # "y" or "n"

plotResults = "y"

# Concentration
conc = "LCC"
run  = "ked"  # "xrf" or "ked"
#sampType = "u" # "u" or "upu"
stage = "s1"  # "s1" or "s2"
# Hardcode sample type
sType = "Pu"   # Options: "U" or "U-Pu"
cat = "aqueous"

if stage == "s1":
    outdir = "s1_cases"
elif stage == "s2":
    outdir = "s2_cases"

# Hardcode file name in for now
specFile = "data/"+run+"_"+conc+"gL_"+sType+".txt"
#specFile = "data/old/300gL_ked_u.txt"

fileName = outdir+"/"+run+"/"+cat+"/hked_v11_"+run+"_"+stage+"_"+conc+"
    gL_"+sType+".inp.o"

fileName = "s1_cases/ked/complex/hked_v12_ked_s1_volox_powder_long.o"
```

```
name = conc+"gL_"+run+"_"+sType
name = "LCC_ked"

# Hardcode figure file names
figName = name + "_full.pdf"
figNameROI = name + "_detail.pdf"
figName2 = name + "_full.png"
figNameROI2 = name + "_detail.png"



# Open and create files needed by this script
# Open the input file for reading
f = open(fileName, "r")

# rename the input file and open it as a post processed output
base = os.path.splitext(fileName)[0]
outFile = base + ".opp"
sourceFile = base + ".src"
if stage == "s2":
    resultsFile = base + ".res"
    r = open(resultsFile, "w")
    r.write(name+",")

# create and open the output files for writing
o = open(outFile, "w")
#s = open(sourceFile, "wb")



# Determine the run type and search terms
```

```
# Tell the user what's going on
print "\nProcessing MCNP output:", fileName
# Search the file for XRF-RUN or KED-RUN tags
with f as search:
    for line in search:
        # Remove '\n' at end of line
        line = line.rstrip()
        if "xrf-s1" in line:
            aType = "XRF-S1"
            print "Run Type:", aType
            break
        elif "xrf-s2" in line:
            aType = "XRF-S2"
            print "Run Type:", aType
            break
        elif "ked-s1" in line:
            aType = "KED-S1"
            print "Run Type:", aType
            break
        elif "ked-s2" in line:
            aType = "KED-S2"
            print "Run Type:", aType
            break

if aType == "XRF-S2" or aType == "KED-S2":
    # Ask the user what to do
    #plotResults = "n" #raw_input("Plot the results? (y/n): ")
    #plotResults = "n"

    # Ask user if a measured spectrum is to be analyzed
```

```
    importSpectrum = "y" #raw_input("Analyze a measured spectrum? (y/n):
        ")
if aType == "XRF-S1" or aType == "KED-S1":
    importSpectrum = "n"
    plotResults = "n"


# Tell the user where the extracted spectrum is
print "MCNP tally extracted to:",outFile

# Set the tally surface/volume search string
if aType == "XRF-S1":
    searchPhrase = " surface 609.2"
    print "Searched MCNP output for:", searchPhrase
elif aType == "XRF-S2":
    searchPhrase = " cell 75"
    print "Searched MCNP output for:", searchPhrase
elif aType == "KED-S1":
    searchPhrase = " surface 313.2"
    print "Searched MCNP output for:", searchPhrase
elif aType == "KED-S2":
    searchPhrase = " cell 44"
    print "Searched MCNP output for:", searchPhrase


# Create an output file and extract spectrum from MCNP output

if aType == "XRF-S1" or aType == "KED-S1":
    # Identify Stage 1 analysis
    print "Extracting Stage 1 F2 tally..."
```

```
    # Search for the specified string in the input file
    with open(fileName) as search:
        for line in search:
            # Remove '\n' at end of line
            line = line.rstrip()
            if searchPhrase in line:
                #for x in range (0,2053):
                #for x in range (0,602):
                #for x in range (0,2051):
                for x in range (0,8194):
                        line = search.next()
                        # Remove the spaces between tally values
                        # and replace with commas
                        line = line.replace('  ','')
                        line = line.replace(' ',',')
                        line = line.replace(' ',',')
                        line = line.replace(',,energy,','energy')
                        line = line.replace(',,total,,',',total,')
                        # Write the data to analysis file in CSV format
                        o.write(line)
    # Close and reopen output file to flush the buffer
    o.close()
    o = open(outFile,"r")


# Create source file from Stage 1 tally for Stage 2 source

# write the Stage 2 SDEF options to the new source file with a Cd-109
    check source
if aType == "XRF-S1":
```

```
    s = open(sourceFile, "wb")
    s2sdef = "SDEF VEC=-0.692 -0.721 0 DIR=1 POS=D1 ERG=FPOS=D2 PAR=2
        ARA=0.001 \n"
    s2sdef = s2sdef + "SI1 L -9.57 -9.97 0   -9.57 -9.97 0 \n"
    s2sdef = s2sdef + "SP1   1               1E-3 \n"
    s2sdef = s2sdef + "DS2 S 4               3 \n"
    s2sdef = s2sdef + "SI3 L 0.02199 0.022163 0.024912 0.02943 0.025455
        0.0880336 \n"
    s2sdef = s2sdef + "SP3 D 0.298  0.561   0.048   0.092  0.0231
        0.0370"
elif aType =="KED-S1":
    s = open(sourceFile, "wb")
    s2sdef = "SDEF VEC=1 0 0 DIR=1 POS=D1 ERG=FPOS=D2 PAR=2 ARA=0.001 \n
        "
    s2sdef = s2sdef + "SI1 L 19.8545 0 0   19.8545 1 0 \n"
    s2sdef = s2sdef + "SP1   1               2E-1 \n"
    s2sdef = s2sdef + "DS2 S 4               3 \n"
    s2sdef = s2sdef + "SI3 L 0.02199 0.022163 0.024912 0.02943 0.025455
        0.0880336 \n"
    s2sdef = s2sdef + "SP3 D 0.298  0.561   0.048   0.092  0.0231
        0.0370"

if aType == "XRF-S1" or aType == "KED-S1":
    print "Writing Stage 2 source file..."
    # write source file header
    sheader = "c Source file processed from output: "
    s.write(sheader + fileName + "\n")
    s.write(s2sdef + "\n")

    # import the source data from the preprocessed output file
```

```
sourcein = np.genfromtxt(outFile, delimiter=",", skip_header=1,
    skip_footer=1,
        usecols=(0,1))
mcnpEnergy = np.genfromtxt(outFile, delimiter=",", skip_header=1,
    skip_footer=1,
        usecols=(0))
mcnpTally = np.genfromtxt(outFile, delimiter=",", skip_header=1,
    skip_footer=1,
        usecols=(1))
mcnpError= np.genfromtxt(outFile, delimiter=",", skip_header=1,
    skip_footer=1,
        usecols=(2))

# sum the source and copy the energies
sumsource = sum(sourcein[:,1])
sourceenergy = sourcein[:,0]

# normalize the Stage 1 tally
for x in range(0,len(sourcein)):
    sourcenorm = sourcein[:,1]/sumsource

# Polynomial fit parameters
#p1 =   5.347e-06
#p2 =   5.042e-06
#p3 =  -3.899e-05
#p4 =  -3.092e-05
#p5 =   8.373e-05
#p6 =   5.541e-05
#p7 =  -5.538e-05
#p8 =   -5.86e-05
```

```
#p9  =  -0.0002095
#p10 = -0.0002799
p1 = -0.2099
p2 =  0.07759
p3 = -0.0003616
p4 = -2.786E-5

# Correct the energy
for i in range(0,len(sourceenergy)):
    sourceenergy[i] = sourceenergy[i] - (p1*sourceenergy[i]**3 + p2*
        sourceenergy[i]**2 + p3*sourceenergy[i] + p4)

# write source energies
for x in range(0,len(sourcein)):
    if x == 0:
        #soutline1 = "SI1 L "
        # Use SI1 L for source with Cd-109
        soutline1 = "SI4 L "
        soutline2 = str(sourceenergy[x])
        s.write(soutline1)
        s.write(soutline2 + "\n")
    else:
        soutline = ("     " + str(sourceenergy[x]))
        s.write(soutline + "\n")

# write the source intensities
for x in range(0,len(sourcein)):
    if x == 0:
        #soutline1 = "SP1 D "
        # Use SP4 D for source with Cd-109
```

```
            soutline1 = "SP4 D "
            soutline2 = str(sourcenorm[x])
            s.write(soutline1)
            s.write(soutline2 + "\n")
        else:
            soutline = ("     " + str(sourcenorm[x]))
            s.write(soutline + "\n")

    # tell user where the source file was written
    print "Stage 1 tally written to Stage 2 source file:", sourceFile
    s.close()

    errorin = np.genfromtxt(outFile, delimiter=",", skip_header=1,
        skip_footer=1,
            usecols=(0,2))


# Import data from Stage 2 runs

if aType == "XRF-S2" or aType == "KED-S2":
    print "Extracting Stage 2 F8 tally..."
    with open(fileName) as search:
        for line in search:
            line = line.rstrip() # remove '\n' at end of line
            if searchPhrase in line:
                for x in range (0,2052):
                    line = search.next()
                    # remove the spaces between tally values
                    # and replace with commas
                    line = line.replace('  ','')
```

```
                line = line.replace(' ',',')
                line = line.replace(' ',',')
                line = line.replace(',,energy,','energy')
                line = line.replace(',,total,,','total,')
                # write the data to analysis file in CSV format
                o.write(line)

# Close and reopen the file to flush the buffer
    o.close()
    o = open(outFile,"r")

# Normalize data from Stage 2 runs

if aType == "XRF-S2" or aType == "KED-S2":
    # import the source data from the preprocessed output file
    spectrumIn = np.genfromtxt(outFile, delimiter=",", skip_header=3,
        skip_footer=1,
            usecols=(0,1))
    #print spectrumIn
    #print "spectrumIn=",len(spectrumIn)

    s2Bins = np.genfromtxt(outFile, delimiter=",", skip_header=3,
        skip_footer=1,
            usecols=(0))
    s2Tally = np.genfromtxt(outFile, delimiter=",", skip_header=3,
        skip_footer=1,
            usecols=(1))
    s2Error = np.genfromtxt(outFile, delimiter=",", skip_header=3,
        skip_footer=1,
            usecols=(2))
```

```
    # Convert Stage 2 bin energies to keV
    # The average offset for the Ka1 and Ka2 peaks is 0.1035 but 0.07
        should
    # make the Kb's align more closely.
    for i in range(0,len(s2Bins)):
        s2Bins[i] = s2Bins[i]*1000-0.07

    # Initialize normalized tally matrix
    s2NormTal = np.zeros((len(s2Tally),1))

    s2CdPeak = "n"
    # Search the S2 tally for the 88 keV peak
    for i in range(0,len(s2Tally)):
        if s2CdPeak == "n":
            if s2Bins[i] > 88.03: #keV
                s2CdPeak = s2Tally[i]
                #s2CdPeak = s2Tally[i] - ((s2Tally[1150]+s2Tally[1171])
                    /2)
                CdPeakLoc = i

    # Normalize the Stage 2 tally
    for i in range(0,len(s2Tally)):
        # Normalize to the Cd-109 peak
        s2NormTal[i] = s2Tally[i]/s2CdPeak
        #s2NormTal[i] = s2Tally[i]/sum(s2Tally)


# K-edge continuum comparison
```

```
if aType == "KED-S2":

    print "\n-=- K-EDGE CONTINUUM ANALYSIS -=-"

    # Set measured lower and upper continuum boundaries
    k_lc_l_model = 1444
    k_lc_u_model = 1517
    k_uc_l_model = 1540
    k_uc_u_model = 1612

    # Measured
    print "\n-=- MODELED CONTINUUM BOUNDS -=-"
    print "Modeled lower K-edge continuum lower bound at channel: ",
        k_lc_l_model,"(",s2Bins[k_lc_l_model]," keV )"
    print "Modeled lower K-edge continuum upper bound at channel: ",
        k_lc_u_model,"(",s2Bins[k_lc_u_model]," keV )"
    print "Modeled upper K-edge continuum lower bound at channel: ",
        k_uc_l_model,"(",s2Bins[k_uc_l_model]," keV )"
    print "Modeled upper K-edge continuum upper bound at channel: ",
        k_uc_u_model,"(",s2Bins[k_uc_u_model]," keV )"

    # Determine continuum area for modeled data
    k_lc_sum_model = np.sum(s2NormTal[k_lc_l_model:k_lc_u_model])
    k_uc_sum_model = np.sum(s2NormTal[k_uc_l_model:k_uc_u_model])

    k_ratio_model = k_lc_sum_model/k_uc_sum_model

    k_ratio_model_unc = np.average(np.average(s2Error[k_lc_l_model:
        k_lc_u_model])+np.average(s2Error[k_uc_l_model:k_uc_u_model]))
```

```
    # Print the results
    print "\n-=- K-EDGE ANALYSIS RESULTS -=-"
    print "Modeled K-edge continuum ratio: ", k_ratio_model
    print "Modeled K-edge continuum uncertainty: ",k_ratio_model_unc

    print "\nK-edge Excel data format"
    print k_ratio_model,k_ratio_model_unc


# Uranium XRF Peak area comparison

if aType == "XRF-S2":

    print "\n-=- URANIUM K ALPHA PEAK AREA ANALYSIS -=-"

    ## Set peak boundaries
    ## U K alpha peaks
    ka2_l = 93.9
    ka2_u = 95.36
    ka1_l = 97.6
    ka1_u = 99.1


    # Set tally channel variables to a string
    ka2_l_chan_t = "n"
    ka2_u_chan_t = "n"
    ka1_l_chan_t = "n"
    ka1_u_chan_t = "n"

    # Tally
```

```
for i in range(0,len(s2Bins)-1):
    if (s2Bins[i] > ka2_l and ka2_l_chan_t == "n"):
        ka2_l_chan_t = i
        print "\nTally Ka2 lower bound at channel: ",ka2_l_chan_t
            ,"(",s2Bins[i]," keV )"
        #print s2NormTal[i]
    elif (s2Bins[i] > ka2_u and ka2_u_chan_t == "n"):
        ka2_u_chan_t = i
        print "Tally Ka2 upper bound at channel: ",ka2_u_chan_t,"(",
            s2Bins[i]," keV )"
        #print s2NormTal[i]
    elif (s2Bins[i] > ka1_l and ka1_l_chan_t == "n"):
        ka1_l_chan_t = i
        print "Tally Ka1 lower bound at channel: ",ka1_l_chan_t,"(",
            s2Bins[i]," keV )"
        #print s2NormTal[i]
    elif (s2Bins[i] > ka1_u and ka1_u_chan_t == "n"):
        ka1_u_chan_t = i
        print "Tally Ka1 upper bound at channel: ",ka1_u_chan_t,"(",
            s2Bins[i]," keV )"
        #print s2NormTal[i]

# Measure the peak areas
ka2_sum_tal = np.sum((s2NormTal[ka2_l_chan_t:ka2_u_chan_t]))

# Measure the peak areas
ka1_sum_tal = np.sum((s2NormTal[ka1_l_chan_t:ka1_u_chan_t]))

# Ka2 calculate the continuum
```

```
ka2_cont_t = ((s2NormTal[ka2_l_chan_t]+s2NormTal[ka2_u_chan_t])/2) *
     (ka2_u_chan_t-ka2_l_chan_t)
# Ka1 calculate the continuum
ka1_cont_t = ((s2NormTal[ka1_l_chan_t]+s2NormTal[ka1_u_chan_t])/2) *
     (ka1_u_chan_t-ka1_l_chan_t)

# Subtract the continuums from the peak areas
ka2_peak_t = ka2_sum_tal - ka2_cont_t
ka1_peak_t = ka1_sum_tal - ka1_cont_t

# U K beta peaks fit as doublets
kb13_l = 109.6
kb13_u = 111.8
kb24_l = 113.6
kb24_u = 116.1

# Set measured channel variables to a string
kb13_l_chan_m = "n"
kb13_u_chan_m = "n"
kb24_l_chan_m = "n"
kb24_u_chan_m = "n"

print "\n-=- URANIUM K BETA PEAK AREA ANALYSIS -=-"

# Set tally channel variables to a string
kb13_l_chan_t = "n"
kb13_u_chan_t = "n"
kb24_l_chan_t = "n"
kb24_u_chan_t = "n"
```

```
# Tally
for i in range(0,len(s2Bins)-1):
    if (s2Bins[i] > kb13_l and kb13_l_chan_t == "n"):
        kb13_l_chan_t = i
        print "\nTally Kb13 lower bound at channel: ",kb13_l_chan_t
            ,"(",s2Bins[i]," keV )"
        #print s2NormTal[i]
    elif (s2Bins[i] > kb13_u and kb13_u_chan_t == "n"):
        kb13_u_chan_t = i
        print "Tally Kb13 upper bound at channel: ",kb13_u_chan_t
            ,"(",s2Bins[i]," keV )"
        #print s2NormTal[i]
    elif (s2Bins[i] > kb24_l and kb24_l_chan_t == "n"):
        kb24_l_chan_t = i
        print "Tally Kb24 lower bound at channel: ",kb24_l_chan_t
            ,"(",s2Bins[i]," keV )"
        #print s2NormTal[i]
    elif (s2Bins[i] > kb24_u and kb24_u_chan_t == "n"):
        kb24_u_chan_t = i
        print "Tally Kb24 upper bound at channel: ",kb24_u_chan_t
            ,"(",s2Bins[i]," keV )"
        #print s2NormTal[i]

# Measure the peak areas
kb13_sum_tal = np.sum((s2NormTal[kb13_l_chan_t:kb13_u_chan_t]))

# Measure the peak areas
kb24_sum_tal = np.sum((s2NormTal[kb24_l_chan_t:kb24_u_chan_t]))

# kb13 calculate the continuum
```

```
kb13_cont_t = ((s2NormTal[kb13_l_chan_t]+s2NormTal[kb13_u_chan_t])
    /2) * (kb13_u_chan_t-kb13_l_chan_t)
# kb24 calculate the continuum
kb24_cont_t = ((s2NormTal[kb24_l_chan_t]+s2NormTal[kb24_u_chan_t])
    /2) * (kb24_u_chan_t-kb24_l_chan_t)

# Subtract the continuums from the peak areas
kb13_peak_t = kb13_sum_tal - kb13_cont_t
kb24_peak_t = kb24_sum_tal - kb24_cont_t



# Calculate the peak uncertainties
# Ka2
#ka2_peak_m_unc = np.sqrt(ka2_peak_m)
#ka2_peak_t_unc = np.sqrt(ka2_peak_t)
# Ka1
#ka1_peak_m_unc = np.sqrt(ka1_peak_m)
#ka1_peak_t_unc = np.sqrt(ka1_peak_t)
# Kb13
#kb13_peak_m_unc = np.sqrt(kb13_peak_m)
#kb13_peak_t_unc = np.sqrt(kb13_peak_t)
# Kb24
#kb24_peak_m_unc = np.sqrt(kb24_peak_m)
#kb24_peak_t_unc = np.sqrt(kb24_peak_t)



# New tally error calculation
ka2_peak_t_unc = np.average(s2Error[ka2_l_chan_t:ka2_u_chan_t])
ka1_peak_t_unc = np.average(s2Error[ka1_l_chan_t:ka1_u_chan_t])
kb13_peak_t_unc = np.average(s2Error[kb13_l_chan_t:kb13_u_chan_t])
```

```
    kb24_peak_t_unc = np.average(s2Error[kb24_l_chan_t:kb24_u_chan_t])

    print "\n-=- URANIUM PEAK AREAS -=-"
    print "Ka2 tally: ",ka2_peak_t
    print "Ka2 tally uncertainty: ",ka2_peak_t_unc

    print "Ka1 tally: ",ka1_peak_t
    print "Ka1 tally uncertainty: ",ka1_peak_t_unc

    print "Kb13 tally: ",kb13_peak_t
    print "Kb13 tally uncertainty: ",kb13_peak_t_unc

    print "Kb24 tally: ",kb24_peak_t
    print "Kb24 tally uncertainty: ",kb24_peak_t_unc

    print "\n"

    # Excel data format
    print "\nUranium Excel data format\n"
    #print ka1_peak_m+","+ka1_peak_m_unc+","+ka2_peak_m+","+
       ka2_peak_m_unc+","+kb13_peak_m+","+kb13_peak_m_unc+","+
       kb24_peak_m+","+kb24_peak_m_unc+","+ka1_peak_t+","+ka1_peak_t_unc
       +","+ka2_peak_t+","+ka2_peak_t_unc+","+kb13_peak_t+","+
       kb13_peak_t_unc+","+kb24_peak_t+","+kb24_peak_t_unc
    print ka1_peak_t,ka1_peak_t_unc,ka2_peak_t,ka2_peak_t_unc,
       kb13_peak_t,kb13_peak_t_unc,kb24_peak_t,kb24_peak_t_unc


# Plutionum XRF peaks
```

```
if aType == "XRF-S2" and sType == "U-Pu":

    print "\n-=- PLUTIONIUM K ALPHA PEAK AREA ANALYSIS -=-"

    # Set peak boundaries
    # Pu K alpha peaks
    ka2_l = 98.9
    ka2_u = 100.1
    ka1_l = 97.8
    ka1_u = 99.0

    # Set measured channel variables to a string
    ka2_l_chan_m = "n"
    ka2_u_chan_m = "n"
    ka1_l_chan_m = "n"
    ka1_u_chan_m = "n"



    # Set tally channel variables to a string
    ka2_l_chan_t = "n"
    ka2_u_chan_t = "n"
    ka1_l_chan_t = "n"
    ka1_u_chan_t = "n"

    # Tally
    for i in range(0,len(s2Bins)-1):
        if (s2Bins[i] > ka2_l and ka2_l_chan_t == "n"):
            ka2_l_chan_t = i
            print "\nTally Ka2 lower bound at channel: ",ka2_l_chan_t
                ,"(",s2Bins[i]," keV )"
```

```
        #print s2NormTal[i]
    elif (s2Bins[i] > ka2_u and ka2_u_chan_t == "n"):
        ka2_u_chan_t = i
        print "Tally Ka2 upper bound at channel: ",ka2_u_chan_t,"(",
            s2Bins[i]," keV )"
        #print s2NormTal[i]
    elif (s2Bins[i] > ka1_l and ka1_l_chan_t == "n"):
        ka1_l_chan_t = i
        print "Tally Ka1 lower bound at channel: ",ka1_l_chan_t,"(",
            s2Bins[i]," keV )"
        #print s2NormTal[i]
    elif (s2Bins[i] > ka1_u and ka1_u_chan_t == "n"):
        ka1_u_chan_t = i
        print "Tally Ka1 upper bound at channel: ",ka1_u_chan_t,"(",
            s2Bins[i]," keV )"
        #print s2NormTal[i]

# Measure the peak areas
ka2_sum_tal = np.sum((s2NormTal[ka2_l_chan_t:ka2_u_chan_t]))

# Measure the peak areas
ka1_sum_tal = np.sum((s2NormTal[ka1_l_chan_t:ka1_u_chan_t]))

# Ka2 calculate the continuum
ka2_cont_t = ((s2NormTal[ka2_l_chan_t]+s2NormTal[ka2_u_chan_t])/2) *
     (ka2_u_chan_t-ka2_l_chan_t)
# Ka1 calculate the continuum
ka1_cont_t = ((s2NormTal[ka1_l_chan_t]+s2NormTal[ka1_u_chan_t])/2) *
     (ka1_u_chan_t-ka1_l_chan_t)
```

```
# Subtract the continuums from the peak areas
ka2_peak_t = ka2_sum_tal - ka2_cont_t
ka1_peak_t = ka1_sum_tal - ka1_cont_t

if ka2_peak_t < 0:
    ka2_cont_t = 0
    ka2_peak_t = 0 #ka2_sum_tal - ka2_cont_t
if ka1_peak_t < 0:
    ka1_cont_t = 0
    ka1_peak_t = 0 #ka1_sum_tal - ka1_cont_t

# Pu K beta peaks fit as doublets
kb13_l = 110.0
kb13_u = 111.6
kb24_l = 114.0
kb24_u = 116.1

# Set measured channel variables to a string
kb13_l_chan_m = "n"
kb13_u_chan_m = "n"
kb24_l_chan_m = "n"
kb24_u_chan_m = "n"

print "\n-=- PLUTONIUM K BETA PEAK AREA ANALYSIS -=-"


# Set tally channel variables to a string
kb13_l_chan_t = "n"
kb13_u_chan_t = "n"
kb24_l_chan_t = "n"
```

```
kb24_u_chan_t = "n"


# Tally
for i in range(0,len(s2Bins)-1):
    if (s2Bins[i] > kb13_l and kb13_l_chan_t == "n"):
        kb13_l_chan_t = i
        print "\nTally Kb13 lower bound at channel: ",kb13_l_chan_t
            ,"(",s2Bins[i]," keV )"
        #print s2NormTal[i]
    elif (s2Bins[i] > kb13_u and kb13_u_chan_t == "n"):
        kb13_u_chan_t = i
        print "Tally Kb13 upper bound at channel: ",kb13_u_chan_t
            ,"(",s2Bins[i]," keV )"
        #print s2NormTal[i]
    elif (s2Bins[i] > kb24_l and kb24_l_chan_t == "n"):
        kb24_l_chan_t = i
        print "Tally Kb24 lower bound at channel: ",kb24_l_chan_t
            ,"(",s2Bins[i]," keV )"
        #print s2NormTal[i]
    elif (s2Bins[i] > kb24_u and kb24_u_chan_t == "n"):
        kb24_u_chan_t = i
        print "Tally Kb24 upper bound at channel: ",kb24_u_chan_t
            ,"(",s2Bins[i]," keV )"
        #print s2NormTal[i]


# Measure the peak areas
kb13_sum_tal = np.sum((s2NormTal[kb13_l_chan_t:kb13_u_chan_t]))


# Measure the peak areas
kb24_sum_tal = np.sum((s2NormTal[kb24_l_chan_t:kb24_u_chan_t]))
```

```
# kb13 calculate the continuum
kb13_cont_t = ((s2NormTal[kb13_l_chan_t]+s2NormTal[kb13_u_chan_t])
    /2) * (kb13_u_chan_t-kb13_l_chan_t)
# kb24 calculate the continuum
kb24_cont_t = ((s2NormTal[kb24_l_chan_t]+s2NormTal[kb24_u_chan_t])
    /2) * (kb24_u_chan_t-kb24_l_chan_t)


# Subtract the continuums from the peak areas
kb13_peak_t = kb13_sum_tal - kb13_cont_t
kb24_peak_t = kb24_sum_tal - kb24_cont_t


if kb13_peak_t < 0:
    kb13_cont_t = 0
    kb13_peak_t = 0 #ka2_sum_tal - ka2_cont_t
if ka1_peak_t < 0:
    kb24_cont_t = 0
    kb24_peak_t = 0 #ka1_sum_tal - ka1_cont_t

# Calculate the peak uncertainties
# Ka2
#ka2_peak_m_unc = np.sqrt(ka2_peak_m)
#ka2_peak_t_unc = np.sqrt(ka2_peak_t)
# Ka1
#ka1_peak_m_unc = np.sqrt(ka1_peak_m)
#ka1_peak_t_unc = np.sqrt(ka1_peak_t)
# Kb13
#kb13_peak_m_unc = np.sqrt(kb13_peak_m)
```

```
#kb13_peak_t_unc = np.sqrt(kb13_peak_t)
# Kb24
#kb24_peak_m_unc = np.sqrt(kb24_peak_m)
#kb24_peak_t_unc = np.sqrt(kb24_peak_t)



# New tally error calculation
ka2_peak_t_unc = np.average(s2Error[ka2_l_chan_t:ka2_u_chan_t])
ka1_peak_t_unc = np.average(s2Error[ka1_l_chan_t:ka1_u_chan_t])
kb13_peak_t_unc = np.average(s2Error[kb13_l_chan_t:kb13_u_chan_t])
kb24_peak_t_unc = np.average(s2Error[kb24_l_chan_t:kb24_u_chan_t])

#ka1_diff = (ka1_peak_m-ka1_peak_t)/ka1_peak_m
#ka2_diff = (ka2_peak_m-ka2_peak_t)/ka2_peak_m
#kb13_diff = (kb13_peak_m-kb13_peak_t)/kb13_peak_m
#kb24_diff = (kb24_peak_m-kb24_peak_t)/kb24_peak_m

print "\n-=- PLUTONIUM PEAK AREAS -=-"

print "Ka2 tally: ",ka2_peak_t
print "Ka2 tally uncertainty: ",ka2_peak_t_unc

print "Ka1 tally: ",ka1_peak_t
print "Ka1 tally uncertainty: ",ka1_peak_t_unc



print "Kb13 tally: ",kb13_peak_t
print "Kb13 tally uncertainty: ",kb13_peak_t_unc

print "Kb24 tally: ",kb24_peak_t
```

```
    print "Kb24 tally uncertainty: ",kb24_peak_t_unc

    #print "\nKa1 error: ",ka1_diff*100," %"
    #print "Ka2 error: ",ka2_diff*100," %"
    #print "Kb13 error: ",kb13_diff*100," %"
    #print "Kb24 error: ",kb24_diff*100," %"
    #print "\n"

    # Excel data format
    print "\nPlutonium Excel data format\n"
    #print ka1_peak_m+","+ka1_peak_m_unc+","+ka2_peak_m+","+
        ka2_peak_m_unc+","+kb13_peak_m+","+kb13_peak_m_unc+","+
        kb24_peak_m+","+kb24_peak_m_unc+","+ka1_peak_t+","+ka1_peak_t_unc
        +","+ka2_peak_t+","+ka2_peak_t_unc+","+kb13_peak_t+","+
        kb13_peak_t_unc+","+kb24_peak_t+","+kb24_peak_t_unc
    print ka1_peak_t,ka1_peak_t_unc,ka2_peak_t,ka2_peak_t_unc,
        kb13_peak_t,kb13_peak_t_unc,kb24_peak_t,kb24_peak_t_unc



# Plot the results

chan = range(0,2048)

# correct the residuals bins by 4 keV
if aType == "KED-S2":
    s2ResidBins = s2Bins[:] - 4
elif aType == "XRF-S2":
    s2ResidBins = s2Bins[:] + 3
```

```
print "\n-=- PLOTTING -=-"
# If plotResults is "y" then call matplotlib
if plotResults == "y":

    # close any previous figures
    plt.close("all")

    # Prompt for plot title
    plotTitle = "" #raw_input("Enter string for plot title: ")

    # Set the figure dimensions
    plt.figure(figsize=(8,4))

    # Plot the tally results
    plt.subplot(1,1,1)
    plt.ylabel("Relative Intensity")
    plt.xlabel("Energy (keV)")
    plt.xlim((0,155))
    #plt.ylim((1E-4,1E2))
    plt.semilogy(s2Bins,s2NormTal,color="black")

    # Set the title
    plt.title(plotTitle)



    #plt.xlim((0,155))
    plt.ylabel("Relative Intensity", fontsize=10)
    # Set the legend location
    plt.legend(loc=1,prop={'size':6})
    plt.tick_params(labelsize=12)
```

```
plt.tick_params(which='both', width=1, labelsize=10)
plt.tick_params(which='major', length=8)
plt.grid(b=True,which="major")
if aType == "XRF-S2":
    if sType == "U":
        # U case
        plt.axvspan(93,100,facecolor='0.5',alpha=0.25)
        plt.axvspan(108,117,facecolor='0.5',alpha=0.25)
    elif sType == "Pu":
        # Pu case
        plt.axvspan(98.5,100,facecolor='g',alpha=0.25)
        plt.axvspan(103,104.5,facecolor='g',alpha=0.25)
        plt.axvspan(115.2,118,facecolor='g',alpha=0.25)
        plt.axvspan(119.5,121.5,facecolor='g',alpha=0.25)
    elif sType == "U-Pu":
        plt.axvspan(99.2,99.9,facecolor='g',alpha=0.25)
        plt.axvspan(103,104.5,facecolor='g',alpha=0.25)
        plt.axvspan(116.5,118,facecolor='g',alpha=0.25)
        plt.axvspan(119.5,121.5,facecolor='g',alpha=0.25)
        plt.axvspan(93,100,facecolor='0.5',alpha=0.25)
        plt.axvspan(109.35,116.2,facecolor='0.5',alpha=0.25)
elif aType == "KED-S2":
    if sType == "U":
        plt.axvspan(114,117,facecolor='0.5',alpha=0.25)
    if sType == "Pu":
        plt.axvspan(119.5,121.5,facecolor='g',alpha=0.25)



# Ask user to save the file
```

```
    #saveFile = "y" #raw_input("Do you want to save full figure as a PDF
        ? (y/n): ")
    if saveFile == "y":
        #figName = raw_input("Enter file name: ")
        plt.savefig(figName, dpi=1000, format='pdf', orientation='
            landscape',
            bbox_inches='tight')
        plt.savefig(figName2, dpi=1000, format='png', orientation='
            landscape',
            bbox_inches='tight')
        print "INFO: Full figures saved"
    elif saveFile =="n":
        print "INFO: Full figures not saved"

#######
# Plot detailed version
# Set the figure dimensions
    plt.figure(figsize=(8,4))

    # Plot the tally results
    plt.subplot(1,1,1)
    plt.ylabel("Relative Intensity")
    plt.xlabel("Energy (keV)")
    #plt.ylim((1E-4,1E2))
    plt.semilogy(s2Bins,s2NormTal,color="black")
    #plt.errorbar(s2Bins,s2Tally,yerr=corrError*s2Tally,color="red",
    #    label="MCNP",errorevery=10)
    ## Set the title
    plt.title(plotTitle)
    plt.xlim((92,122))
```

```
plt.ylabel("Relative Intensity", fontsize=10)
# Set the legend location
plt.legend(loc=1,prop={'size':6})
plt.tick_params(labelsize=12)
plt.tick_params(which='both', width=1, labelsize=10)
plt.tick_params(which='major', length=8)
plt.grid(b=True,which="major")
if aType == "XRF-S2":
    if sType == "U":
        # U case
        plt.axvspan(93,100,facecolor='0.5',alpha=0.25)
        plt.axvspan(108,117,facecolor='0.5',alpha=0.25)
    elif sType == "Pu":
        # Pu case
        plt.axvspan(98.5,100,facecolor='g',alpha=0.25)
        plt.axvspan(103,104.5,facecolor='g',alpha=0.25)
        plt.axvspan(115.2,118,facecolor='g',alpha=0.25)
        plt.axvspan(119.5,121.5,facecolor='g',alpha=0.25)
    elif sType == "U-Pu":
        plt.axvspan(99.2,99.9,facecolor='g',alpha=0.25)
        plt.axvspan(103,104.5,facecolor='g',alpha=0.25)
        plt.axvspan(116.5,118,facecolor='g',alpha=0.25)
        plt.axvspan(119.5,121.5,facecolor='g',alpha=0.25)
        plt.axvspan(93,100,facecolor='0.5',alpha=0.25)
        plt.axvspan(109.35,116.2,facecolor='0.5',alpha=0.25)
elif aType == "KED-S2":
    if sType == "U":
        plt.axvspan(114,117,facecolor='0.5',alpha=0.25)
    if sType == "Pu":
        plt.axvspan(119.5,121.5,facecolor='g',alpha=0.25)
```

```
    # Ask user to save the file
    #saveFile = "y" #raw_input("Do you want to save zoomed figure as a
        PDF? (y/n): ")
    if saveFile == "y":
        #figNameROI = raw_input("Enter file name: ")
        plt.savefig(figNameROI, dpi=1000, format='pdf', orientation='
            landscape',
            bbox_inches='tight')
        plt.savefig(figNameROI2, dpi=1000, format='png', orientation='
            landscape',
            bbox_inches='tight')
        print "INFO: Detailed figures saved"
    elif saveFile =="n":
        print "INFO: Detailed figures not saved"

## Print message if not plotting results
elif plotResults == "n":
    print "INFO: Results not plotted"


plt.show()



# close all open files
f.close()
o.close()

if stage == "s2":
    r.close()
```

```
#s.close()
```

### C.1.5 Active Samples

```
# -*- coding: utf-8 -*-
# HPAT: HKED Python Analysis Tool Pyro
# Version 2.0
#   Matthew T. Cook
# 12 March 2014
# Department of Nuclear Engineering
# University of Tennessee, Knoxville

# import modules
import os as os
import numpy as np
import matplotlib.pyplot as plt
import warnings



# Warning control
# Set script to ignore runtime warnings from modelResid divide by zero
warnings.simplefilter("ignore")



# Clear the screen
os.system('clear')

# Identify script
print " ---------------------------------"
print "| HKED Python Analysis Tool v. 2.0|"
print " ---------------------------------"
```

```
# Run in automatic
auto = "n" # "y" or "n"

# Save plot files?
saveFile = "y" # "y" or "n"

plotResults = "y"

# Concentration
conc = "250"
run  = "ked"  # "xrf" or "ked"
#sampType = "u" # "u" or "upu"
stage = "s2"  # "s1" or "s2"
# Hardcode sample type
sType = "U-Th"  # Options: "U" or "U-Th"
ratio = ""
cat = "gamma"

if stage == "s1":
    outdir = "s1_cases"
elif stage == "s2":
    outdir = "s2_cases"

# Hardcode file name in for now
specFile = "data/"+run+"_"+conc+"gL_"+sType+".txt"
#specFile = "data/old/300gL_ked_u.txt"

fileName = outdir+"/"+run+"/"+cat+"/hked_v11_"+run+"_"+stage+"_"+conc+"
    gL_"+sType+ratio+".inp.o"
```

```
fileName = "s2_cases/xrf/gamma/hked_v12_xrf_s2_gamma_45GWd_T_subtracted.
    o"


name = conc+"gL_"+run+"_"+sType+"_"+ratio

name = "45mwd_t_xrf_subtracted"

# Hardcode figure file names
figName = name + "_full.pdf"
figNameROI = name + "_detail.pdf"
figName2 = name + "_full.png"
figNameROI2 = name + "_detail.png"


# Open and create files needed by this script
# Open the input file for reading
f = open(fileName, "r")

# rename the input file and open it as a post processed output
base = os.path.splitext(fileName)[0]
outFile = base + ".opp"
sourceFile = base + ".src"
if stage == "s2":
    resultsFile = "RawResults_KED_surrogates_ratio.txt"
    r = open(resultsFile, "a")
    r.write(name)
    r.write(",")
```

```
# create and open the output files for writing
o = open(outFile, "w")
#s = open(sourceFile, "wb")


# Determine the run type and search terms

# Tell the user what's going on
print "\nProcessing MCNP output:", fileName
# Search the file for XRF-RUN or KED-RUN tags
with f as search:
    for line in search:
        # Remove '\n' at end of line
        line = line.rstrip()
        if "xrf-s1" in line:
            aType = "XRF-S1"
            print "Run Type:", aType
            break
        elif "xrf-s2" in line:
            aType = "XRF-S2"
            print "Run Type:", aType
            break
        elif "ked-s1" in line:
            aType = "KED-S1"
            print "Run Type:", aType
            break
        elif "ked-s2" in line:
            aType = "KED-S2"
            print "Run Type:", aType
            break
```

```
if aType == "XRF-S2" or aType == "KED-S2":
    # Ask the user what to do
    #plotResults = "n" #raw_input("Plot the results? (y/n): ")
    #plotResults = "n"

    # Ask user if a measured spectrum is to be analyzed
    importSpectrum = "y" #raw_input("Analyze a measured spectrum? (y/n):
        ")
if aType == "XRF-S1" or aType == "KED-S1":
    importSpectrum = "n"
    plotResults = "n"


# Tell the user where the extracted spectrum is
print "MCNP tally extracted to:",outFile

# Set the tally surface/volume search string
if aType == "XRF-S1":
    searchPhrase = " surface 609.2"
    print "Searched MCNP output for:", searchPhrase
elif aType == "XRF-S2":
    searchPhrase = " cell 75"
    print "Searched MCNP output for:", searchPhrase
elif aType == "KED-S1":
    searchPhrase = " surface 313.2"
    print "Searched MCNP output for:", searchPhrase
elif aType == "KED-S2":
    searchPhrase = " cell 44"
    print "Searched MCNP output for:", searchPhrase
```

```
# Create an output file and extract spectrum from MCNP output

if aType == "XRF-S1" or aType == "KED-S1":
    # Identify Stage 1 analysis
    print "Extracting Stage 1 F2 tally..."
    # Search for the specified string in the input file
    with open(fileName) as search:
        for line in search:
            # Remove '\n' at end of line
            line = line.rstrip()
            if searchPhrase in line:
                #for x in range (0,2053):
                #for x in range (0,602):
                #for x in range (0,2051):
                for x in range (0,8194):
                        line = search.next()
                        # Remove the spaces between tally values
                        # and replace with commas
                        line = line.replace('  ','')
                        line = line.replace(' ',',')
                        line = line.replace(' ',',')
                        line = line.replace(',,energy,','energy')
                        line = line.replace(',,total,,',',total,')
                        # Write the data to analysis file in CSV format
                        o.write(line)
    # Close and reopen output file to flush the buffer
    o.close()
    o = open(outFile,"r")
```

```
# Create source file from Stage 1 tally for Stage 2 source

# write the Stage 2 SDEF options to the new source file with a Cd-109
    check source
if aType == "XRF-S1":
    s = open(sourceFile, "wb")
    s2sdef = "SDEF VEC=-0.857167 -0.515038 0 DIR=1 POS=D1 ERG=FPOS=D2
        PAR=2 ARA=0.0071\n"
    s2sdef = s2sdef + "SI1 L 3.60 -2.07 0   -9.57 -9.97 0 \n"
    s2sdef = s2sdef + "SP1   1              1E-3 \n"
    s2sdef = s2sdef + "DS2 S 4              3 \n"
    s2sdef = s2sdef + "SI3 L 0.02199 0.022163 0.024912 0.02943 0.025455
        0.0880336 \n"
    s2sdef = s2sdef + "SP3 D 0.298  0.561   0.048   0.092  0.0231
        0.0370"
elif aType =="KED-S1":
    s = open(sourceFile, "wb")
    s2sdef = "SDEF VEC=1 0 0 DIR=1 POS=D1 ERG=FPOS=D2 PAR=2 ARA=0.001 \n
        "
    s2sdef = s2sdef + "SI1 L 11.65 0 0   19.8545 0 0 \n"
    s2sdef = s2sdef + "SP1   1              2E-1 \n"
    s2sdef = s2sdef + "DS2 S 4              3 \n"
    s2sdef = s2sdef + "SI3 L 0.02199 0.022163 0.024912 0.02943 0.025455
        0.0880336 \n"
    s2sdef = s2sdef + "SP3 D 0.298  0.561   0.048   0.092  0.0231
        0.0370"

if aType == "XRF-S1" or aType == "KED-S1":
```

```
print "Writing Stage 2 source file..."
# write source file header
sheader = "c Source file processed from output: "
s.write(sheader + fileName + "\n")
s.write(s2sdef + "\n")

# import the source data from the preprocessed output file
sourcein = np.genfromtxt(outFile, delimiter=",", skip_header=1,
    skip_footer=1,
        usecols=(0,1))
mcnpEnergy = np.genfromtxt(outFile, delimiter=",", skip_header=1,
    skip_footer=1,
        usecols=(0))
mcnpTally = np.genfromtxt(outFile, delimiter=",", skip_header=1,
    skip_footer=1,
        usecols=(1))
mcnpError= np.genfromtxt(outFile, delimiter=",", skip_header=1,
    skip_footer=1,
        usecols=(2))

# sum the source and copy the energies
sumsource = sum(sourcein[:,1])
sourceenergy = sourcein[:,0]

# normalize the Stage 1 tally
for x in range(0,len(sourcein)):
    sourcenorm = sourcein[:,1]/sumsource

# Polynomial fit parameters
#p1  =   5.347e-06
```

```
#p2  =   5.042e-06
#p3  =  -3.899e-05
#p4  =  -3.092e-05
#p5  =   8.373e-05
#p6  =   5.541e-05
#p7  =  -5.538e-05
#p8  =   -5.86e-05
#p9  =  -0.0002095
#p10 = -0.0002799
p1 = -0.2099
p2 =  0.07759
p3 = -0.0003616
p4 = -2.786E-5

# Correct the energy
for i in range(0,len(sourceenergy)):
    sourceenergy[i] = sourceenergy[i] - (p1*sourceenergy[i]**3 + p2*
        sourceenergy[i]**2 + p3*sourceenergy[i] + p4)

# write source energies
for x in range(0,len(sourcein)):
    if x == 0:
        #soutline1 = "SI1 L "
        # Use SI1 L for source with Cd-109
        soutline1 = "SI4 L "
        soutline2 = str(sourceenergy[x])
        s.write(soutline1)
        s.write(soutline2 + "\n")
    else:
        soutline = ("     " + str(sourceenergy[x]))
```

```
            s.write(soutline + "\n")

    # write the source intensities
    for x in range(0,len(sourcein)):
        if x == 0:
            #soutline1 = "SP1 D "
            # Use SP4 D for source with Cd-109
            soutline1 = "SP4 D "
            soutline2 = str(sourcenorm[x])
            s.write(soutline1)
            s.write(soutline2 + "\n")
        else:
            soutline = ("     " + str(sourcenorm[x]))
            s.write(soutline + "\n")

    # tell user where the source file was written
    print "Stage 1 tally written to Stage 2 source file:", sourceFile
    s.close()

    errorin = np.genfromtxt(outFile, delimiter=",", skip_header=1,
        skip_footer=1,
            usecols=(0,2))


# Import data from Stage 2 runs

if aType == "XRF-S2" or aType == "KED-S2":
    print "Extracting Stage 2 F8 tally..."
    with open(fileName) as search:
        for line in search:
```

```
            line = line.rstrip() # remove '\n' at end of line
            if searchPhrase in line:
                for x in range (0,2052):
                    line = search.next()
                    # remove the spaces between tally values
                    # and replace with commas
                    line = line.replace('  ','')
                    line = line.replace(' ',',')
                    line = line.replace(' ',',')
                    line = line.replace(',,energy,','energy')
                    line = line.replace(',,total,,',',total,')
                    # write the data to analysis file in CSV format
                    o.write(line)

# Close and reopen the file to flush the buffer
    o.close()
    o = open(outFile,"r")

# Normalize data from Stage 2 runs

if aType == "XRF-S2" or aType == "KED-S2":
    # import the source data from the preprocessed output file
    spectrumIn = np.genfromtxt(outFile, delimiter=",", skip_header=3,
        skip_footer=1,
            usecols=(0,1))
    #print spectrumIn
    #print "spectrumIn=",len(spectrumIn)

    s2Bins = np.genfromtxt(outFile, delimiter=",", skip_header=3,
        skip_footer=1,
```

```
        usecols=(0))
s2Tally = np.genfromtxt(outFile, delimiter=",", skip_header=3,
    skip_footer=1,
        usecols=(1))
s2Error = np.genfromtxt(outFile, delimiter=",", skip_header=3,
    skip_footer=1,
        usecols=(2))

# Convert Stage 2 bin energies to keV
# The average offset for the Ka1 and Ka2 peaks is 0.1035 but 0.07
    should
# make the Kb's align more closely.
for i in range(0,len(s2Bins)):
    s2Bins[i] = s2Bins[i]*1000-0.07

# Initialize normalized tally matrix
s2NormTal = np.zeros((len(s2Tally),1))

s2CdPeak = "n"
# Search the S2 tally for the 88 keV peak
for i in range(0,len(s2Tally)):
    if s2CdPeak == "n":
        if s2Bins[i] > 88.03: #keV
            s2CdPeak = s2Tally[i]
            #s2CdPeak = s2Tally[i] - ((s2Tally[1150]+s2Tally[1171])
                /2)
            CdPeakLoc = i

# Normalize the Stage 2 tally
for i in range(0,len(s2Tally)):
```

```
        # Normalize to the Cd-109 peak
        s2NormTal[i] = s2Tally[i]/s2CdPeak
        #s2NormTal[i] = s2Tally[i]/sum(s2Tally)


# K-edge continuum comparison

if aType == "KED-S2":

    print "\n-=- K-EDGE CONTINUUM ANALYSIS -=-"

    # Set measured lower and upper continuum boundaries
    k_lc_l_model = 1444
    k_lc_u_model = 1517
    k_uc_l_model = 1540
    k_uc_u_model = 1612

    # Measured
    print "\n-=- MODELED CONTINUUM BOUNDS -=-"
    print "Modeled lower K-edge continuum lower bound at channel: ",
        k_lc_l_model,"(",s2Bins[k_lc_l_model]," keV )"
    print "Modeled lower K-edge continuum upper bound at channel: ",
        k_lc_u_model,"(",s2Bins[k_lc_u_model]," keV )"
    print "Modeled upper K-edge continuum lower bound at channel: ",
        k_uc_l_model,"(",s2Bins[k_uc_l_model]," keV )"
    print "Modeled upper K-edge continuum upper bound at channel: ",
        k_uc_u_model,"(",s2Bins[k_uc_u_model]," keV )"

    # Determine continuum area for modeled data
    k_lc_sum_model = np.sum(s2NormTal[k_lc_l_model:k_lc_u_model])
```

```
    k_uc_sum_model = np.sum(s2NormTal[k_uc_l_model:k_uc_u_model])

    k_ratio_model = k_lc_sum_model/k_uc_sum_model

    k_ratio_model_unc = np.average(np.average(s2Error[k_lc_l_model:
        k_lc_u_model])+np.average(s2Error[k_uc_l_model:k_uc_u_model]))

    # Print the results
    print "\n-=- K-EDGE ANALYSIS RESULTS -=-"
    print "Modeled K-edge continuum ratio: ", k_ratio_model
    print "Modeled K-edge continuum uncertainty: ",k_ratio_model_unc

    print "\nK-edge Excel data format"
    print k_ratio_model,k_ratio_model_unc
    rout = str(k_ratio_model)+","+str(k_ratio_model_unc)+"\n"
    r.write(rout)


# Uranium XRF Peak area comparison

if aType == "XRF-S2":

    print "\n-=- URANIUM K ALPHA PEAK AREA ANALYSIS -=-"

    ## Set peak boundaries
    ## U K alpha peaks
    ka2_l = 93.9
    ka2_u = 95.36
    ka1_l = 97.6
    ka1_u = 99.1
```

```
# Set tally channel variables to a string
ka2_l_chan_t = "n"
ka2_u_chan_t = "n"
ka1_l_chan_t = "n"
ka1_u_chan_t = "n"

# Tally
for i in range(0,len(s2Bins)-1):
    if (s2Bins[i] > ka2_l and ka2_l_chan_t == "n"):
        ka2_l_chan_t = i
        print "\nTally Ka2 lower bound at channel: ",ka2_l_chan_t
            ,"(",s2Bins[i]," keV )"
        #print s2NormTal[i]
    elif (s2Bins[i] > ka2_u and ka2_u_chan_t == "n"):
        ka2_u_chan_t = i
        print "Tally Ka2 upper bound at channel: ",ka2_u_chan_t,"(",
            s2Bins[i]," keV )"
        #print s2NormTal[i]
    elif (s2Bins[i] > ka1_l and ka1_l_chan_t == "n"):
        ka1_l_chan_t = i
        print "Tally Ka1 lower bound at channel: ",ka1_l_chan_t,"(",
            s2Bins[i]," keV )"
        #print s2NormTal[i]
    elif (s2Bins[i] > ka1_u and ka1_u_chan_t == "n"):
        ka1_u_chan_t = i
        print "Tally Ka1 upper bound at channel: ",ka1_u_chan_t,"(",
            s2Bins[i]," keV )"
        #print s2NormTal[i]
```

```
# Measure the peak areas
ka2_sum_tal = np.sum((s2NormTal[ka2_l_chan_t:ka2_u_chan_t]))

# Measure the peak areas
ka1_sum_tal = np.sum((s2NormTal[ka1_l_chan_t:ka1_u_chan_t]))

# Ka2 calculate the continuum
ka2_cont_t = ((s2NormTal[ka2_l_chan_t]+s2NormTal[ka2_u_chan_t])/2) *
     (ka2_u_chan_t-ka2_l_chan_t)
# Ka1 calculate the continuum
ka1_cont_t = ((s2NormTal[ka1_l_chan_t]+s2NormTal[ka1_u_chan_t])/2) *
     (ka1_u_chan_t-ka1_l_chan_t)

# Subtract the continuums from the peak areas
ka2_peak_t = ka2_sum_tal - ka2_cont_t
ka1_peak_t = ka1_sum_tal - ka1_cont_t

# U K beta peaks fit as doublets
kb13_l = 109.6
kb13_u = 111.8
kb24_l = 113.6
kb24_u = 116.1

# Set measured channel variables to a string
kb13_l_chan_m = "n"
kb13_u_chan_m = "n"
kb24_l_chan_m = "n"
kb24_u_chan_m = "n"
```

```
print "\n-=- URANIUM K BETA PEAK AREA ANALYSIS -=-"

# Set tally channel variables to a string
kb13_l_chan_t = "n"
kb13_u_chan_t = "n"
kb24_l_chan_t = "n"
kb24_u_chan_t = "n"

# Tally
for i in range(0,len(s2Bins)-1):
    if (s2Bins[i] > kb13_l and kb13_l_chan_t == "n"):
        kb13_l_chan_t = i
        print "\nTally Kb13 lower bound at channel: ",kb13_l_chan_t
            ,"(",s2Bins[i]," keV )"
        #print s2NormTal[i]
    elif (s2Bins[i] > kb13_u and kb13_u_chan_t == "n"):
        kb13_u_chan_t = i
        print "Tally Kb13 upper bound at channel: ",kb13_u_chan_t
            ,"(",s2Bins[i]," keV )"
        #print s2NormTal[i]
    elif (s2Bins[i] > kb24_l and kb24_l_chan_t == "n"):
        kb24_l_chan_t = i
        print "Tally Kb24 lower bound at channel: ",kb24_l_chan_t
            ,"(",s2Bins[i]," keV )"
        #print s2NormTal[i]
    elif (s2Bins[i] > kb24_u and kb24_u_chan_t == "n"):
        kb24_u_chan_t = i
        print "Tally Kb24 upper bound at channel: ",kb24_u_chan_t
            ,"(",s2Bins[i]," keV )"
        #print s2NormTal[i]
```

```
# Measure the peak areas
kb13_sum_tal = np.sum((s2NormTal[kb13_l_chan_t:kb13_u_chan_t]))

# Measure the peak areas
kb24_sum_tal = np.sum((s2NormTal[kb24_l_chan_t:kb24_u_chan_t]))

# kb13 calculate the continuum
kb13_cont_t = ((s2NormTal[kb13_l_chan_t]+s2NormTal[kb13_u_chan_t])
    /2) * (kb13_u_chan_t-kb13_l_chan_t)
# kb24 calculate the continuum
kb24_cont_t = ((s2NormTal[kb24_l_chan_t]+s2NormTal[kb24_u_chan_t])
    /2) * (kb24_u_chan_t-kb24_l_chan_t)

# Subtract the continuums from the peak areas
kb13_peak_t = kb13_sum_tal - kb13_cont_t
kb24_peak_t = kb24_sum_tal - kb24_cont_t



# Calculate the peak uncertainties
# Ka2
#ka2_peak_m_unc = np.sqrt(ka2_peak_m)
#ka2_peak_t_unc = np.sqrt(ka2_peak_t)
# Ka1
#ka1_peak_m_unc = np.sqrt(ka1_peak_m)
#ka1_peak_t_unc = np.sqrt(ka1_peak_t)
# Kb13
#kb13_peak_m_unc = np.sqrt(kb13_peak_m)
#kb13_peak_t_unc = np.sqrt(kb13_peak_t)
# Kb24
```

```
#kb24_peak_m_unc = np.sqrt(kb24_peak_m)
#kb24_peak_t_unc = np.sqrt(kb24_peak_t)



# New tally error calculation
ka2_peak_t_unc = np.average(s2Error[ka2_l_chan_t:ka2_u_chan_t])
ka1_peak_t_unc = np.average(s2Error[ka1_l_chan_t:ka1_u_chan_t])
kb13_peak_t_unc = np.average(s2Error[kb13_l_chan_t:kb13_u_chan_t])
kb24_peak_t_unc = np.average(s2Error[kb24_l_chan_t:kb24_u_chan_t])

print "\n-=- URANIUM PEAK AREAS -=-"
print "Ka2 tally: ",ka2_peak_t
print "Ka2 tally uncertainty: ",ka2_peak_t_unc

print "Ka1 tally: ",ka1_peak_t
print "Ka1 tally uncertainty: ",ka1_peak_t_unc

print "Kb13 tally: ",kb13_peak_t
print "Kb13 tally uncertainty: ",kb13_peak_t_unc

print "Kb24 tally: ",kb24_peak_t
print "Kb24 tally uncertainty: ",kb24_peak_t_unc

print "\n"

# Excel data format
print "\nUranium Excel data format\n"
```

```
    rout = str(ka1_peak_t)+","+str(ka1_peak_t_unc)+","+str(ka2_peak_t)
        +","+str(ka2_peak_t_unc)+","+str(kb13_peak_t)+","+str(
        kb13_peak_t_unc)+","+str(kb24_peak_t)+","+str(kb24_peak_t_unc)
        +","
    print ka1_peak_t,ka1_peak_t_unc,ka2_peak_t,ka2_peak_t_unc,
        kb13_peak_t,kb13_peak_t_unc,kb24_peak_t,kb24_peak_t_unc
    r.write(rout)


# Plutionum XRF peaks

if aType == "XRF-S2" and sType == "U-Th":

    print "\n-=- THORIUM K ALPHA PEAK AREA ANALYSIS -=-"

    # Set peak boundaries
    # Thorium K alpha peaks
    ka2_l = 89.2
    ka2_u = 90.4
    ka1_l = 92.8
    ka1_u = 93.8

    # Set measured channel variables to a string
    ka2_l_chan_m = "n"
    ka2_u_chan_m = "n"
    ka1_l_chan_m = "n"
    ka1_u_chan_m = "n"


    # Set tally channel variables to a string
```

```
ka2_l_chan_t = "n"
ka2_u_chan_t = "n"
ka1_l_chan_t = "n"
ka1_u_chan_t = "n"

# Tally
for i in range(0,len(s2Bins)-1):
    if (s2Bins[i] > ka2_l and ka2_l_chan_t == "n"):
        ka2_l_chan_t = i
        print "\nTally Ka2 lower bound at channel: ",ka2_l_chan_t
            ,"(",s2Bins[i]," keV )"
        #print s2NormTal[i]
    elif (s2Bins[i] > ka2_u and ka2_u_chan_t == "n"):
        ka2_u_chan_t = i
        print "Tally Ka2 upper bound at channel: ",ka2_u_chan_t,"(",
            s2Bins[i]," keV )"
        #print s2NormTal[i]
    elif (s2Bins[i] > ka1_l and ka1_l_chan_t == "n"):
        ka1_l_chan_t = i
        print "Tally Ka1 lower bound at channel: ",ka1_l_chan_t,"(",
            s2Bins[i]," keV )"
        #print s2NormTal[i]
    elif (s2Bins[i] > ka1_u and ka1_u_chan_t == "n"):
        ka1_u_chan_t = i
        print "Tally Ka1 upper bound at channel: ",ka1_u_chan_t,"(",
            s2Bins[i]," keV )"
        #print s2NormTal[i]

# Measure the peak areas
ka2_sum_tal = np.sum((s2NormTal[ka2_l_chan_t:ka2_u_chan_t]))
```

```
# Measure the peak areas
ka1_sum_tal = np.sum((s2NormTal[ka1_l_chan_t:ka1_u_chan_t]))

# Ka2 calculate the continuum
ka2_cont_t = ((s2NormTal[ka2_l_chan_t]+s2NormTal[ka2_u_chan_t])/2) *
     (ka2_u_chan_t-ka2_l_chan_t)
# Ka1 calculate the continuum
ka1_cont_t = ((s2NormTal[ka1_l_chan_t]+s2NormTal[ka1_u_chan_t])/2) *
     (ka1_u_chan_t-ka1_l_chan_t)

# Subtract the continuums from the peak areas
ka2_peak_t = ka2_sum_tal - ka2_cont_t
ka1_peak_t = ka1_sum_tal - ka1_cont_t

if ka2_peak_t < 0:
    ka2_cont_t = 0
    ka2_peak_t = 0 #ka2_sum_tal - ka2_cont_t
if ka1_peak_t < 0:
    ka1_cont_t = 0
    ka1_peak_t = 0 #ka1_sum_tal - ka1_cont_t

# Thorium K beta peaks fit as doublets
kb13_l = 103.98
kb13_u = 106.55
kb24_l = 114.0
kb24_u = 116.1

# Set measured channel variables to a string
kb13_l_chan_m = "n"
```

```
kb13_u_chan_m = "n"
kb24_l_chan_m = "n"
kb24_u_chan_m = "n"

print "\n-=- THORIUM K BETA PEAK AREA ANALYSIS -=-"



# Set tally channel variables to a string
kb13_l_chan_t = "n"
kb13_u_chan_t = "n"
kb24_l_chan_t = "n"
kb24_u_chan_t = "n"

# Tally
for i in range(0,len(s2Bins)-1):
    if (s2Bins[i] > kb13_l and kb13_l_chan_t == "n"):
        kb13_l_chan_t = i
        print "\nTally Kb13 lower bound at channel: ",kb13_l_chan_t
            ,"(",s2Bins[i]," keV )"
        #print s2NormTal[i]
    elif (s2Bins[i] > kb13_u and kb13_u_chan_t == "n"):
        kb13_u_chan_t = i
        print "Tally Kb13 upper bound at channel: ",kb13_u_chan_t
            ,"(",s2Bins[i]," keV )"
        #print s2NormTal[i]
    elif (s2Bins[i] > kb24_l and kb24_l_chan_t == "n"):
        kb24_l_chan_t = i
        print "Tally Kb24 lower bound at channel: ",kb24_l_chan_t
            ,"(",s2Bins[i]," keV )"
        #print s2NormTal[i]
```

```
    elif (s2Bins[i] > kb24_u and kb24_u_chan_t == "n"):
        kb24_u_chan_t = i
        print "Tally Kb24 upper bound at channel: ",kb24_u_chan_t
            ,"(",s2Bins[i]," keV )"
        #print s2NormTal[i]

# Measure the peak areas
kb13_sum_tal = np.sum((s2NormTal[kb13_l_chan_t:kb13_u_chan_t]))

# Measure the peak areas
kb24_sum_tal = np.sum((s2NormTal[kb24_l_chan_t:kb24_u_chan_t]))

# kb13 calculate the continuum
kb13_cont_t = ((s2NormTal[kb13_l_chan_t]+s2NormTal[kb13_u_chan_t])
    /2) * (kb13_u_chan_t-kb13_l_chan_t)
# kb24 calculate the continuum
kb24_cont_t = ((s2NormTal[kb24_l_chan_t]+s2NormTal[kb24_u_chan_t])
    /2) * (kb24_u_chan_t-kb24_l_chan_t)


# Subtract the continuums from the peak areas
kb13_peak_t = kb13_sum_tal - kb13_cont_t
kb24_peak_t = kb24_sum_tal - kb24_cont_t


if kb13_peak_t < 0:
    kb13_cont_t = 0
    kb13_peak_t = 0 #ka2_sum_tal - ka2_cont_t
if ka1_peak_t < 0:
    kb24_cont_t = 0
```

```
    kb24_peak_t = 0 #ka1_sum_tal - ka1_cont_t

# Calculate the peak uncertainties
# Ka2
#ka2_peak_m_unc = np.sqrt(ka2_peak_m)
#ka2_peak_t_unc = np.sqrt(ka2_peak_t)
# Ka1
#ka1_peak_m_unc = np.sqrt(ka1_peak_m)
#ka1_peak_t_unc = np.sqrt(ka1_peak_t)
# Kb13
#kb13_peak_m_unc = np.sqrt(kb13_peak_m)
#kb13_peak_t_unc = np.sqrt(kb13_peak_t)
# Kb24
#kb24_peak_m_unc = np.sqrt(kb24_peak_m)
#kb24_peak_t_unc = np.sqrt(kb24_peak_t)



# New tally error calculation
ka2_peak_t_unc = np.average(s2Error[ka2_l_chan_t:ka2_u_chan_t])
ka1_peak_t_unc = np.average(s2Error[ka1_l_chan_t:ka1_u_chan_t])
kb13_peak_t_unc = np.average(s2Error[kb13_l_chan_t:kb13_u_chan_t])
kb24_peak_t_unc = np.average(s2Error[kb24_l_chan_t:kb24_u_chan_t])

#ka1_diff = (ka1_peak_m-ka1_peak_t)/ka1_peak_m
#ka2_diff = (ka2_peak_m-ka2_peak_t)/ka2_peak_m
#kb13_diff = (kb13_peak_m-kb13_peak_t)/kb13_peak_m
#kb24_diff = (kb24_peak_m-kb24_peak_t)/kb24_peak_m

print "\n-=- THORIUM PEAK AREAS -=-"
```

```
print "Ka2 tally: ",ka2_peak_t
print "Ka2 tally uncertainty: ",ka2_peak_t_unc

print "Ka1 tally: ",ka1_peak_t
print "Ka1 tally uncertainty: ",ka1_peak_t_unc



print "Kb13 tally: ",kb13_peak_t
print "Kb13 tally uncertainty: ",kb13_peak_t_unc

print "Kb24 tally: ",kb24_peak_t
print "Kb24 tally uncertainty: ",kb24_peak_t_unc

#print "\nKa1 error: ",ka1_diff*100," %"
#print "Ka2 error: ",ka2_diff*100," %"
#print "Kb13 error: ",kb13_diff*100," %"
#print "Kb24 error: ",kb24_diff*100," %"
#print "\n"

# Excel data format
print "\nThorium Excel data format\n"
#rout = ka1_peak_t+","+ka1_peak_t_unc+","+ka2_peak_t+","+
    ka2_peak_t_unc+","+kb13_peak_t+","+kb13_peak_t_unc+","+
    kb24_peak_t+","+kb24_peak_t_unc
#print ka1_peak_t,ka1_peak_t_unc,ka2_peak_t,ka2_peak_t_unc,
    kb13_peak_t,kb13_peak_t_unc,kb24_peak_t,kb24_peak_t_unc
```

```
    rout = str(ka1_peak_t)+","+str(ka1_peak_t_unc)+","+str(ka2_peak_t)
        +","+str(ka2_peak_t_unc)+","+str(kb13_peak_t)+","+str(
        kb13_peak_t_unc)+","+str(kb24_peak_t)+","+str(kb24_peak_t_unc)+"\
        n"
    print ka1_peak_t,ka1_peak_t_unc,ka2_peak_t,ka2_peak_t_unc,
        kb13_peak_t,kb13_peak_t_unc,kb24_peak_t,kb24_peak_t_unc
    r.write(rout)


# Plot the results

chan = range(0,2048)

# correct the residuals bins by 4 keV
if aType == "KED-S2":
    s2ResidBins = s2Bins[:] - 4
elif aType == "XRF-S2":
    s2ResidBins = s2Bins[:] + 3

print "\n-=- PLOTTING -=-"
# If plotResults is "y" then call matplotlib
if plotResults == "y":

    # close any previous figures
    plt.close("all")

    # Prompt for plot title
    plotTitle = "" #raw_input("Enter string for plot title: ")

    # Set the figure dimensions
```

```
plt.figure(figsize=(8,4))

# Plot the tally results
plt.subplot(1,1,1)
plt.ylabel("Relative Intensity")
plt.xlabel("Energy (keV)", fontsize=10)

plt.xlim((0,155))
#plt.ylim((1E-4,1E2))
plt.semilogy(s2Bins,s2NormTal,color="black")

# Set the title
plt.title(plotTitle)



#plt.xlim((0,155))
plt.ylabel("Relative Intensity", fontsize=10)
# Set the legend location
plt.legend(loc=1,prop={'size':6})
plt.tick_params(labelsize=12)
plt.tick_params(which='both', width=1, labelsize=10)
plt.tick_params(which='major', length=8)
plt.grid(b=True,which="major")
if aType == "XRF-S2":
    if sType == "U":
        # U case
        plt.axvspan(93,100,facecolor='0.5',alpha=0.25)
        plt.axvspan(108,117,facecolor='0.5',alpha=0.25)
    elif sType == "U-Th":
        # U-Th case
```

```
            plt.axvspan(93.85,99.53,facecolor='0.5',alpha=0.25)
            plt.axvspan(109.5,116.2,facecolor='0.5',alpha=0.25)
            #plt.axvspan(89.2,93.8,facecolor='g',alpha=0.25)
            #plt.axvspan(104.04,109.45,facecolor='g',alpha=0.25)
    elif aType == "KED-S2":
        plt.axvspan(114,117,facecolor='0.5',alpha=0.25)



    # Ask user to save the file
    #saveFile = "y" #raw_input("Do you want to save full figure as a PDF
        ? (y/n): ")
    if saveFile == "y":
        #figName = raw_input("Enter file name: ")
        plt.savefig(figName, dpi=1000, format='pdf', orientation='
            landscape',
            bbox_inches='tight')
        plt.savefig(figName2, dpi=1000, format='png', orientation='
            landscape',
            bbox_inches='tight')
        print "INFO: Full figures saved"
    elif saveFile =="n":
        print "INFO: Full figures not saved"

#######
# Plot detailed version
# Set the figure dimensions
    plt.figure(figsize=(8,4))

    # Plot the tally results
    plt.subplot(1,1,1)
```

```
plt.ylabel("Relative Intensity")
#plt.ylim((1E-4,1E2))
plt.semilogy(s2Bins,s2NormTal,color="black")
#plt.errorbar(s2Bins,s2Tally,yerr=corrError*s2Tally,color="red",
#    label="MCNP",errorevery=10)
## Set the title
plt.title(plotTitle)
plt.xlim((88,117))
plt.ylabel("Relative Intensity", fontsize=10)
plt.xlabel("Energy (keV)", fontsize=10)

# Set the legend location
plt.legend(loc=1,prop={'size':6})
plt.tick_params(labelsize=12)
plt.tick_params(which='both', width=1, labelsize=10)
plt.tick_params(which='major', length=8)
plt.grid(b=True,which="major")
if aType == "XRF-S2":
    if sType == "U":
        # U case
        plt.axvspan(93,100,facecolor='0.5',alpha=0.25)
        plt.axvspan(108,117,facecolor='0.5',alpha=0.25)
    elif sType == "U-Th":
        # U-Th case
        plt.axvspan(93.85,99.53,facecolor='0.5',alpha=0.25)
        plt.axvspan(109.5,116.2,facecolor='0.5',alpha=0.25)
        #plt.axvspan(89.2,93.8,facecolor='g',alpha=0.25)
        #plt.axvspan(104.04,109.45,facecolor='g',alpha=0.25)
elif aType == "KED-S2":
    plt.axvspan(114,117,facecolor='0.5',alpha=0.25)
```

```
    # Ask user to save the file
    #saveFile = "y" #raw_input("Do you want to save zoomed figure as a
        PDF? (y/n): ")
    if saveFile == "y":
        #figNameROI = raw_input("Enter file name: ")
        plt.savefig(figNameROI, dpi=1000, format='pdf', orientation='
            landscape',
            bbox_inches='tight')
        plt.savefig(figNameROI2, dpi=1000, format='png', orientation='
            landscape',
            bbox_inches='tight')
        print "INFO: Detailed figures saved"
    elif saveFile =="n":
        print "INFO: Detailed figures not saved"

## Print message if not plotting results
elif plotResults == "n":
    print "INFO: Results not plotted"


plt.show()



# close all open files
f.close()
o.close()

if stage == "s2":
    r.close()
```

```
#s.close()
```

## C.2 Hybrid K-edge Results Plotting Script

```
# HKED Results Processing Script
# Matthew T. Cook
# 10 February 2015

# Import needed modules
import matplotlib.pyplot as plt
import numpy as np

# Specify the data file
resultsPath = "ready_results/"
dataFile = resultsPath + "Results_XRF_U-Pu.csv"
aType = "XRF" # KED or XRF
sType = "U-Pu" # U or U-Pu

# Read in data from results files
runID = np.genfromtxt(dataFile,delimiter=",", usecols=(0),dtype="string
    ")

# Read in data based on type of run
if aType == "KED":
    dataIn = np.genfromtxt(dataFile, delimiter=",", usecols
        =(1,2,3,4,5,6,7,8))
elif aType == "XRF" and sType == "U":
    dataIn = np.genfromtxt(dataFile, delimiter=",", usecols
        =(1,2,3,4,5,6,7,8,9,
                                                    10,11,12,13,14,15,
```

```
                                                        16,17,18,19,20))
elif aType == "XRF" and sType == "U-Pu":
    dataIn = np.genfromtxt(dataFile, delimiter=",", usecols
        =(1,2,3,4,5,6,7,8,9,
                                                        10,11,12,13,14,15,

                                                        16,17,18,19,20,21,

                                                        22,23,24,25,26,27,

                                                        28,29,30,31,32,33,

                                                        34,35,36))

# Plot the K-edge results
if aType == "KED":

    # Close open plots
    plt.close("all")

    # Uranium only samples
    if sType == "U":
        # Plot the continuum ratios vs. concentration
        #plt.figure(figsize=(8,4))
        fig = plt.figure(figsize=(8,4))
        fig.canvas.set_window_title("Uranium KED")
        # Set axis limits
        plt.xlim((0,350))
        plt.ylim((0,5))
        # Plot options
```

```
    plt.ylabel("Continuum Ratio (Arb.)")
    plt.xlabel("Concentration (g/L)")
    plt.grid(b=True,which="major")
    # Measured data
    plt.scatter(dataIn[:,0],dataIn[:,4],color="blue",label="
        Experimental",marker="o")
    plt.errorbar(dataIn[:,0],dataIn[:,4],yerr=dataIn[:,5],color="blue
        ",linestyle="none")
    # Modeled data
    plt.scatter(dataIn[:,0],dataIn[:,6],color="red",label="MCNP",
        marker="^")
    plt.errorbar(dataIn[:,0],dataIn[:,6],yerr=dataIn[:,7],color="red
        ",linestyle="none")
    # Show the legend
    plt.legend(loc=2,prop={'size':8})
    plt.show()
    # Save the figures
    plt.savefig("KED_U.pdf", dpi=1000, format='pdf', orientation='
        landscape',bbox_inches='tight')
    plt.savefig("KED_U.png", dpi=1000, format='png', orientation='
        landscape',bbox_inches='tight')

# Mixed samples
if sType == "U-Pu":
    # Plot the continuum ratios vs. concentration
    #plt.figure(figsize=(8,4))
    fig = plt.figure(figsize=(8,4))
    fig.canvas.set_window_title("Uranium/Plutonium KED")
    # Plot options
    plt.ylabel("Continuum Ratio (Arb.)")
```

```
        plt.xlabel("Concentration (g/L)")
        plt.grid(b=True,which="major")
        # Measured data
        plt.scatter(dataIn[:,0],dataIn[:,2],color="blue",label="
            Experimental",marker="o")
        plt.errorbar(dataIn[:,0],dataIn[:,2],yerr=dataIn[:,3],color="blue
            ",linestyle="none")
        # Modeled data
        plt.scatter(dataIn[:,0],dataIn[:,4],color="red",label="MCNP",
            marker="^")
        plt.errorbar(dataIn[:,0],dataIn[:,4],yerr=dataIn[:,5],color="red
            ",linestyle="none")
        # Show the legend
        plt.legend(loc=2,prop={'size':8})
        plt.show()
        # Save the figures
        plt.savefig("KED_U-Pu.pdf", dpi=1000, format='pdf', orientation='
            landscape',bbox_inches='tight')
        plt.savefig("KED_U-Pu.png", dpi=1000, format='png', orientation='
            landscape',bbox_inches='tight')

elif aType == "XRF":

    # Close open plots
    plt.close("all")

    # Uranium only samples
    if sType == "U":
        # Plot the Ka1 intensity vs. concentration
        #plt.figure(figsize=(8,4))
```

```
fig = plt.figure(figsize=(8,4))
fig.canvas.set_window_title("Uranium K_alpha1")
# Plot options
plt.ylabel(r"K$_{\alpha1}$ Peak Intensity (Arb.)")
plt.xlabel("Concentration (g/L)")
# Axis limits
plt.ylim((-5,450))
plt.xlim((-5,350))
plt.grid(b=True,which="major")
# Measured data
plt.scatter(dataIn[:,0],dataIn[:,4],color="blue",label="
    Experimental",marker="o")
plt.errorbar(dataIn[:,0],dataIn[:,4],yerr=dataIn[:,5],color="blue
    ",linestyle="none")
# Modeled data
plt.scatter(dataIn[:,0],dataIn[:,12],color="red",label="MCNP",
    marker="^")
plt.errorbar(dataIn[:,0],dataIn[:,12],yerr=dataIn[:,13],color="
    red",linestyle="none")
# Show the legend
plt.legend(loc=2,prop={'size':8})
# Save the figures
plt.savefig("XRF_U_Ka1_U.pdf", dpi=1000, format='pdf',
    orientation='landscape',bbox_inches='tight')
plt.savefig("XRF_U_Ka1_U.png", dpi=1000, format='png',
    orientation='landscape',bbox_inches='tight')

# Plot the Ka2 intensity vs. concentration
#plt.figure(figsize=(8,4))
fig = plt.figure(figsize=(8,4))
```

```
fig.canvas.set_window_title("Uranium K_alpha2")
# Plot options
plt.ylabel(r"K$_{\alpha2}$ Peak Intensity (Arb.)")
plt.xlabel("Concentration (g/L)")
# Axis limits
plt.ylim((-5,250))
plt.xlim((-5,350))
plt.grid(b=True,which="major")
# Measured data
plt.scatter(dataIn[:,0],dataIn[:,6],color="blue",label="
    Experimental",marker="o")
plt.errorbar(dataIn[:,0],dataIn[:,6],yerr=dataIn[:,7],color="blue
    ",linestyle="none")
# Modeled data
plt.scatter(dataIn[:,0],dataIn[:,14],color="red",label="MCNP",
    marker="^")
plt.errorbar(dataIn[:,0],dataIn[:,14],yerr=dataIn[:,15],color="
    red",linestyle="none")
# Show the legend
plt.legend(loc=2,prop={'size':8})
# Save the figures
plt.savefig("XRF_U_Ka2_U.pdf", dpi=1000, format='pdf',
    orientation='landscape',bbox_inches='tight')
plt.savefig("XRF_U_Ka2_U.png", dpi=1000, format='png',
    orientation='landscape',bbox_inches='tight')

# Plot the Kb13 intensity vs. concentration
#plt.figure(figsize=(8,4))
fig = plt.figure(figsize=(8,4))
fig.canvas.set_window_title("Uranium K_beta13")
```

```
# Plot options
plt.ylabel(r"K$_{\beta13}$ Peak Intensity (Arb.)")
plt.xlabel("Concentration (g/L)")
# Axis limits
plt.ylim((-5,160))
plt.xlim((-5,350))
plt.grid(b=True,which="major")
# Measured data
plt.scatter(dataIn[:,0],dataIn[:,8],color="blue",label="
    Experimental",marker="o")
plt.errorbar(dataIn[:,0],dataIn[:,8],yerr=dataIn[:,9],color="blue
    ",linestyle="none")
# Modeled data
plt.scatter(dataIn[:,0],dataIn[:,16],color="red",label="MCNP",
    marker="^")
plt.errorbar(dataIn[:,0],dataIn[:,16],yerr=dataIn[:,17],color="
    red",linestyle="none")
# Show the legend
plt.legend(loc=2,prop={'size':8})
# Save the figures
plt.savefig("XRF_U_Kb13_U.pdf", dpi=1000, format='pdf',
    orientation='landscape',bbox_inches='tight')
plt.savefig("XRF_U_Kb13_U.png", dpi=1000, format='png',
    orientation='landscape',bbox_inches='tight')

# Plot the Kb24 intensity vs. concentration
#plt.figure(figsize=(8,4))
fig = plt.figure(figsize=(8,4))
fig.canvas.set_window_title("Uranium K_beta24")
# Plot options
```

```
    plt.ylabel(r"K$_{\beta24}$ Peak Intensity (Arb.)")
    plt.xlabel("Concentration (g/L)")
    # Axis limits
    plt.ylim((-5,60))
    plt.xlim((-5,350))
    plt.grid(b=True,which="major")
    # Measured data
    plt.scatter(dataIn[:,0],dataIn[:,10],color="blue",label="
        Experimental",marker="o")
    plt.errorbar(dataIn[:,0],dataIn[:,10],yerr=dataIn[:,11],color="
        blue",linestyle="none")
    # Modeled data
    plt.scatter(dataIn[:,0],dataIn[:,18],color="red",label="MCNP",
        marker="^")
    plt.errorbar(dataIn[:,0],dataIn[:,18],yerr=dataIn[:,19],color="
        red",linestyle="none")
    # Show the legend
    plt.legend(loc=2,prop={'size':8})
    plt.show()
    # Save the figures
    plt.savefig("XRF_U_Kb24_U.pdf", dpi=1000, format='pdf',
        orientation='landscape',bbox_inches='tight')
    plt.savefig("XRF_U_Kb24_U.png", dpi=1000, format='png',
        orientation='landscape',bbox_inches='tight')

# Mixed samples
if sType == "U-Pu":
    # Uranium peaks
    # Plot the U Ka1 intensity vs. concentration
    plt.figure(figsize=(8,4))
```

```
# Plot options
plt.ylabel(r"K$_{\alpha1}$ Peak Intensity (Arb.)")
plt.xlabel("Concentration (g/L)")
plt.grid(b=True,which="major")
# Measured data
plt.scatter(dataIn[:,0],dataIn[:,4],color="blue",label="
    Experimental",marker="o")
plt.errorbar(dataIn[:,0],dataIn[:,4],yerr=dataIn[:,5],color="blue
    ",linestyle="none")
# Modeled data
plt.scatter(dataIn[:,0],dataIn[:,12],color="red",label="MCNP",
    marker="^")
plt.errorbar(dataIn[:,0],dataIn[:,12],yerr=dataIn[:,13],color="
    red",linestyle="none")
# Show the legend
plt.legend(loc=2,prop={'size':8})
# Save the figures
plt.savefig("XRF_U_Ka1_U-Pu.pdf", dpi=1000, format='pdf',
    orientation='landscape',bbox_inches='tight')
plt.savefig("XRF_U_Ka1_U-Pu.png", dpi=1000, format='png',
    orientation='landscape',bbox_inches='tight')

# Plot the U Ka2 intensity vs. concentration
plt.figure(figsize=(8,4))
# Plot options
plt.ylabel(r"K$_{\alpha2}$ Peak Intensity (Arb.)")
plt.xlabel("Concentration (g/L)")
plt.grid(b=True,which="major")
# Measured data
```

```
plt.scatter(dataIn[:,0],dataIn[:,6],color="blue",label="
    Experimental",marker="o")
plt.errorbar(dataIn[:,0],dataIn[:,6],yerr=dataIn[:,7],color="blue
    ",linestyle="none")
# Modeled data
plt.scatter(dataIn[:,0],dataIn[:,14],color="red",label="MCNP",
    marker="^")
plt.errorbar(dataIn[:,0],dataIn[:,14],yerr=dataIn[:,15],color="
    red",linestyle="none")
# Show the legend
plt.legend(loc=2,prop={'size':8})
# Save the figures
plt.savefig("XRF_U_Ka2_U-Pu.pdf", dpi=1000, format='pdf',
    orientation='landscape',bbox_inches='tight')
plt.savefig("XRF_U_Ka2_U-Pu.png", dpi=1000, format='png',
    orientation='landscape',bbox_inches='tight')

# Plot the U Kb13 intensity vs. concentration
plt.figure(figsize=(8,4))
# Plot options
plt.ylabel(r"K$_{\beta13}$ Peak Intensity (Arb.)")
plt.xlabel("Concentration (g/L)")
plt.grid(b=True,which="major")
# Measured data
plt.scatter(dataIn[:,0],dataIn[:,8],color="blue",label="
    Experimental",marker="o")
plt.errorbar(dataIn[:,0],dataIn[:,8],yerr=dataIn[:,9],color="blue
    ",linestyle="none")
# Modeled data
```

```
plt.scatter(dataIn[:,0],dataIn[:,16],color="red",label="MCNP",
    marker="^")
plt.errorbar(dataIn[:,0],dataIn[:,16],yerr=dataIn[:,17],color="
    red",linestyle="none")
# Show the legend
plt.legend(loc=2,prop={'size':8})
# Save the figures
plt.savefig("XRF_U_Kb13_U-Pu.pdf", dpi=1000, format='pdf',
    orientation='landscape',bbox_inches='tight')
plt.savefig("XRF_U_Kb13_U-Pu.png", dpi=1000, format='png',
    orientation='landscape',bbox_inches='tight')

# Plot the U Kb24 intensity vs. concentration
plt.figure(figsize=(8,4))
# Plot options
plt.ylabel(r"K$_{\beta24}$ Peak Intensity (Arb.)")
plt.xlabel("Concentration (g/L)")
plt.grid(b=True,which="major")
# Measured data
plt.scatter(dataIn[:,0],dataIn[:,10],color="blue",label="
    Experimental",marker="o")
plt.errorbar(dataIn[:,0],dataIn[:,10],yerr=dataIn[:,11],color="
    blue",linestyle="none")
# Modeled data
plt.scatter(dataIn[:,0],dataIn[:,18],color="red",label="MCNP",
    marker="^")
plt.errorbar(dataIn[:,0],dataIn[:,18],yerr=dataIn[:,19],color="
    red",linestyle="none")
# Show the legend
plt.legend(loc=2,prop={'size':8})
```

```
# Save the figures
plt.savefig("XRF_U_Kb24_U-Pu.pdf", dpi=1000, format='pdf',
    orientation='landscape',bbox_inches='tight')
plt.savefig("XRF_U_Kb24_U-Pu.png", dpi=1000, format='png',
    orientation='landscape',bbox_inches='tight')

# Plutonium Peaks
# Plot the Pu Ka1 intensity vs. concentration
plt.figure(figsize=(8,4))
## Set axis limits
#plt.xlim((100,275))
#plt.ylim((90,300))
# Plot options
plt.ylabel(r"Pu K$_{\alpha1}$ Peak Intensity (Arb.)")
plt.xlabel("Concentration (g/L)")
plt.grid(b=True,which="major")
# Measured data
plt.scatter(dataIn[:,2],dataIn[:,20],color="blue",label="
    Experimental",marker="o")
plt.errorbar(dataIn[:,2],dataIn[:,20],yerr=dataIn[:,21],color="
    blue",linestyle="none")
# Modeled data
plt.scatter(dataIn[:,2],dataIn[:,28],color="red",label="MCNP",
    marker="^")
plt.errorbar(dataIn[:,2],dataIn[:,28],yerr=dataIn[:,29],color="
    red",linestyle="none")
# Show the legend
plt.legend(loc=2,prop={'size':8})
# Save the figures
```

```
plt.savefig("XRF_Pu_Ka1_U-Pu.pdf", dpi=1000, format='pdf',
    orientation='landscape',bbox_inches='tight')
plt.savefig("XRF_Pu_Ka1_U-Pu.png", dpi=1000, format='png',
    orientation='landscape',bbox_inches='tight')

# Plot the Pu Ka2 intensity vs. concentration
plt.figure(figsize=(8,4))
## Set axis limits
#plt.xlim((100,275))
#plt.ylim((50,200))
# Plot options
plt.ylabel(r"Pu K$_{\alpha2}$ Peak Intensity (Arb.)")
plt.xlabel("Concentration (g/L)")
plt.grid(b=True,which="major")
# Measured data
plt.scatter(dataIn[:,2],dataIn[:,22],color="blue",label="
    Experimental",marker="o")
plt.errorbar(dataIn[:,2],dataIn[:,22],yerr=dataIn[:,23],color="
    blue",linestyle="none")
# Modeled data
plt.scatter(dataIn[:,2],dataIn[:,30],color="red",label="MCNP",
    marker="^")
plt.errorbar(dataIn[:,2],dataIn[:,30],yerr=dataIn[:,31],color="
    red",linestyle="none")
# Show the legend
plt.legend(loc=2,prop={'size':8})
# Save the figures
plt.savefig("XRF_Pu_Ka2_U-Pu.pdf", dpi=1000, format='pdf',
    orientation='landscape',bbox_inches='tight')
```

```
plt.savefig("XRF_Pu_Ka2_U-Pu.png", dpi=1000, format='png',
    orientation='landscape',bbox_inches='tight')


# Plot the Pu Kb13 intensity vs. concentration
plt.figure(figsize=(8,4))
## Set axis limits
#plt.xlim((100,275))
#plt.ylim((10,40))
# Plot options
plt.ylabel(r"Pu K$_{\beta13}$ Peak Intensity (Arb.)")
plt.xlabel("Concentration (g/L)")
plt.grid(b=True,which="major")
# Measured data
plt.scatter(dataIn[:,2],dataIn[:,24],color="blue",label="
    Experimental",marker="o")
plt.errorbar(dataIn[:,2],dataIn[:,24],yerr=dataIn[:,25],color="
    blue",linestyle="none")
# Modeled data
plt.scatter(dataIn[:,2],dataIn[:,32],color="red",label="MCNP",
    marker="^")
plt.errorbar(dataIn[:,2],dataIn[:,32],yerr=dataIn[:,33],color="
    red",linestyle="none")
# Show the legend
plt.legend(loc=2,prop={'size':8})
# Save the figures
plt.savefig("XRF_Pu_Kb13_U-Pu.pdf", dpi=1000, format='pdf',
    orientation='landscape',bbox_inches='tight')
plt.savefig("XRF_Pu_Kb13_U-Pu.png", dpi=1000, format='png',
    orientation='landscape',bbox_inches='tight')
```

```
# Plot the Pu Kb24 intensity vs. concentration
plt.figure(figsize=(8,4))
## Set axis limits
#plt.xlim((100,275))
#plt.ylim((90,300))
# Plot options
plt.ylabel(r"Pu K$_{\beta24}$ Peak Intensity (Arb.)")
plt.xlabel("Concentration (g/L)")
plt.grid(b=True,which="major")
# Measured data
plt.scatter(dataIn[:,2],dataIn[:,26],color="blue",label="
    Experimental",marker="o")
plt.errorbar(dataIn[:,2],dataIn[:,26],yerr=dataIn[:,27],color="
    blue",linestyle="none")
# Modeled data
plt.scatter(dataIn[:,2],dataIn[:,34],color="red",label="MCNP",
    marker="^")
plt.errorbar(dataIn[:,2],dataIn[:,34],yerr=dataIn[:,35],color="
    red",linestyle="none")
# Show the legend
plt.legend(loc=2,prop={'size':8})
# Save the figures
plt.savefig("XRF_Pu_Kb24_U-Pu.pdf", dpi=1000, format='pdf',
    orientation='landscape',bbox_inches='tight')
plt.savefig("XRF_Pu_Kb24_U-Pu.png", dpi=1000, format='png',
    orientation='landscape',bbox_inches='tight')

# Show the plots
```

```
        plt.show()
```

## C.3 X-ray Spectrum Stretch Tool (X2ST)

```
# X-ray spectrum stretcher


import numpy as np


# Import spectrum data
importSpec = "mxr-160_stretch2.txt"
f = open(importSpec,"r")
specIn = np.genfromtxt(importSpec,delimiter=",")
# Open output file
outFile = "mxr-160_stretch3.txt"
o = open(outFile,"w")
#p = open("xrSpec_out.txt","w")



# Initialize output array
specOut = np.zeros((2*len(specIn),2))
xrSpec = np.zeros((len(specOut),2))
binwdth = 0.0125/2
avg = np.zeros((len(specIn)-1,1))

## Calculate averages between bins
#for i in range(0,len(avg)):
#    avg[i] = (specIn[i,1]+specIn[i+1,1])/2



j = 0
```

```
# Generate the energy bins
for i in range(0,len(specOut)):
    if i == 0:
        specOut[i,0] = 15
        #specOut[i,1] = specIn[i,1]
    else:
        specOut[i,0] = specOut[i-1,0]+binwdth/2
        #specOut[i+1,1] = specIn[i-j,1]

# Generate counts
for i in range(0,len(specIn)):
    specOut[j,1] = specIn[i,1]
    j = j+2

# Take averages
for i in range(0,len(specOut)-1):
    if specOut[i,1] == 0:
        specOut[i,1] = (specOut[i-1,1]+specOut[i+1,1])/2


# Write to file
for i in range(0,len(specOut)):
    # Write to file
    o.write(str(specOut[i,0])+","+str(specOut[i,1]) +"\n")


for i in range(0,len(specOut)):
    xrSpec[i,0] = specOut[i,0]/1000
    xrSpec[i,1] = specOut[i,1]/sum(specOut[:,1])
```

```
## Write to file
#for i in range(0,len(xrSpec)):
#    # Write to file
#    #p.write(str(xrSpec[i,0])+","+str(xrSpec[i,1]) +"\n")
#    p.write("     "+str(xrSpec[i,0])+"\n")
#
#
#for i in range(0,len(xrSpec)):
#    # Write to file
#    #p.write(str(xrSpec[i,0])+","+str(xrSpec[i,1]) +"\n")
#    p.write("     "+str(xrSpec[i,1])+"\n")

print specOut



# Stretch the spectrum

# Close all files
f.close()
o.close()
#p.close()
```

# Vita

Matthew Cook was born in 1985 Knoxville, Tennessee to Gary and Angela Cook. He has one sister, Jennifer. He went to Shannondale Elementary, Halls Middle, and Halls High Schools developing an interest in science and engineering. He spent his undergraduate years working in information securty at the University of Tennessee gaining an interest in security related topics. Cook earned his Bachelor's degree in 2009 in Nuclear Engineering and decided to continue on into graduate school. His Master's research involved modeling nuclear weapons effects, specifically fallout. The culmination of his Master's work was developing and testing a computer program to model the fallout generated in water-surface nuclear explosions. After completion of the Master's degree he taught undergraduate radiation detection laboratory sections for three semesters. Cook then started his PhD work in modeling a hybrid k-edge densitometer to support extension of the measurement technique to new nuclear reprocessing applications. During his PhD research he also worked on a variety of other projects including synthetic nuclear melt glass development and analysis, graduate radiation detection and radiochemical course development, and computational infrastructure development and upkeep. He completed his Ph.D. work in 2015 and upon graduation plans to pursue a career in nuclear security, safeguards, and forensics.